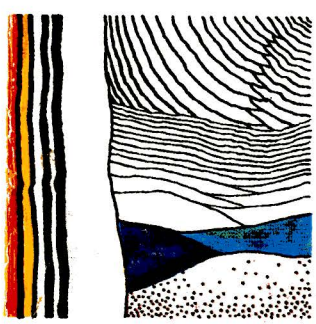
Edouard Labin

Comprendre l'informatique



Bordas initiation

Peu après sa sortie de l'Ecole Supérieure d'Electricité de Paris, Edouard Labin devient directeur d'un laboratoire de recherches dans l'industrie électronique. En 1943, il lance et anime, à la Faculté des sciences de Buenos Aires, un enseignement supérieur d'électronique.
De retour en France, il étudie les tubes à vide spéciaux, puis fonde un département d'ordinateurs. Il devient ensuite directeur technique d'une société de conseils spécialisée dans l'étude de systèmes et s'intéresse à divers domaines scientifiques tels que l'informatique, la recherche opérationnelle et l'économie. En 1960, Edouard Labin s'oriente vers les méthodes modernes d'enseignement, d'information et de documentation, ce que les Américains appellent « l'industrie de la connaissance ».
Détenteur de vingt-six brevets internationaux dans les domaines de l'électronique et de la pédagogie, Edouard Labin a publié plusieurs ouvrages, ainsi que quarante articles dans les revues savantes du monde entier. L'étendue de ses connaissances générales, l'expérience qu'il a acquise en matière pédagogique, jointes à un style aisé et imagé, le prédisposaient à la rédaction de textes scientifiques de haute vulgarisation.
C'est ce qui lui a valu de recevoir des Editions Bordas, en 1971, la mission de lancer et de diriger la présente collection.

BORDAS-INITIATION
collection dirigée
par Édouard Labin

Série A :
Sciences de la nature et techniques
Volume 3

COMPRENDRE L'INFORMATIQUE

par
Édouard Labin

Bordas
Paris-Montréal-Bruxelles

Dans la même collection BORDAS-INITIATION

COMPRENDRE LES MATHÉMATIQUES
par E. Labin

COMPRENDRE L'ÉLECTRONIQUE
par E. Labin
en préparation.

Ouvrages du même auteur

ÉLECTROMAGNÉTISME
(250 pages en supplément à une traduction de Max Planck)
Alcan, 1939, épuisé.

ÉNERGIE ATOMIQUE
(ouvrage de vulgarisation en langue espagnole)
Espasa Calpe, 1950.

LE CALCUL OPÉRATIONNEL
Masson, 1952, épuisé.

QUANTOS
les mathématiques fondamentales en « programmé »
12 volumes, Bordas, 1969, 2e édition en cours.

Les Éditions BORDAS rappellent qu'aux Éditions J. Delmas et Cie, a paru un ouvrage intitulé également *Comprendre l'informatique,* dans la collection « Ce qu'il vous faut savoir », mais dont le contenu et l'esprit sont différents de celui que nous vous présentons.

© Bordas 1973 - N° d'éditeur 177 731 211
D 1973 0190 324
ISBN 2-04-000025-9

« Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de l'auteur, ou de ses ayants-droit ou ayants-cause, est illicite (loi du 11 mars 1957, alinéa 1er de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal. La loi du 11 mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste, et non destinées à une utilisation collective d'une part, et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration ».

Printed in France

« Bientôt notre cerveau deviendra inutile; les ordinateurs penseront pour nous, et nous n'aurons plus qu'à digérer. »

« A quand le mariage par ordinateur ? »

« L'homme va enfin réussir à se transformer en robot. »

« Bah! En quoi la conquête de la Lune peut-elle m'intéresser, c'est une affaire d'ordinateurs. »

« N'appelez plus votre médecin, appelez un IBM 360. »

Etc.

Des phrases de ce genre courent partout.

Si, après avoir lu ce livre,
vous comprenez combien elles sont fausses,
vous n'aurez pas perdu votre temps.

Premier survol

On fait des fiches

L'arme la plus puissante de l'homme : le « codage »

Du cri à la mémoire

On traite l'information

Au cœur de tout : le « programme »

Voici ce que fait l'ordinateur

Que veut-on dire quand on dit qu'une machine « comprend » une information ?

L'ordinateur est un esclave

Articulation de l'ouvrage

On fait des fiches

« Gai, gai, marions-nous! »

Et si nous le faisions par ordinateur?

Qu'est-ce que ça pourrait bien signifier que de se marier par ordinateur?

Pour le comprendre, voyons d'abord comment on se marie par une agence.

Le premier soin d'un monsieur, par exemple, qui s'adresse à une agence matrimoniale pour convoler, est de se décrire lui-même et de décrire la partenaire idéale qu'il souhaite. Comment « décrit-on » une personne? Celle-ci se caractérise par les valeurs que prennent chez elle une série de *paramètres* physiques, caractériels, sociaux. L'agence établit donc deux fiches, l'une (*D*) décrivant le demandeur, l'autre (*R*) décrivant la recherchée.

Ces fiches *D* et *R* auront, par exemple, l'aspect suivant :

	Nom des paramètres	Valeur du paramètre chez le demandeur *D*	Valeur du paramètre chez la recherchée *R*
physiques ou corporels	âge	42 ans	25 ans
	tour de taille	120 cm	80 cm
	tour de poitrine	90 cm	105 cm
	aspect général	épais mais sympathique	très belle et sportive
	couleur des yeux	gris trottoir	vert jade
	couleur des cheveux	brun	blond doré
caractériels	patience	moyenne	infinie
	affectuosité	grande	extrême
	fidélité	bonne	totale
	sensibilité	50 %	100 %
sociaux	famille	épicerie	haute banque
	profession	placier	pianiste
	culture	B.S.T.U.J.V.K.	licenciée ès Lettres
	revenus	10 000 F par mois	nuls

Vous avez bien compris qu'en ce qui concerne les paramètres de la recherchée, l'agence note ceux *souhaités* par le demandeur. La fiche *D* du demandeur décrit un être présent, la fiche *R* de la recherchée décrit un être encore imaginaire. Vous admettrez sans peine de dire que les deux fiches, *D* et *R*, contiennent des **informations** sur les deux êtres. Voilà le grand mot lâché, le premier mot-clé de notre promenade, celui qui a donné naissance au nom même du domaine que nous parcourons : l'**informatique.**

Nous reviendrons à loisir sur la nature exacte de ces **informations,** et apprendrons en particulier à bien cerner le genre de celles qu'aiment les ordinateurs. Pour l'instant, suivons pas à pas les opérations que va accomplir notre agence matrimoniale. Après avoir établi les deux fiches, *D* et *R*, sur du bristol, elle les range à côté de toutes celles qu'elle possède déjà concernant d'autres candidats au mariage. Vous savez qu'en général ce rangement s'effectue dans ces fameuses boîtes de bois ou de carton où les fiches se tiennent debout les unes derrière les autres. C'est une disposition universellement adoptée parce qu'elle permet *d'**extraire** et d'**introduire*** facilement les fiches individuelles quand on en a besoin. Pour faciliter la consultation du fichier, l'agence

aura pris diverses dispositions intelligentes, comme de donner à chaque fiche un numéro, de distinguer les fiches des demandeurs de celles des recherchés en leur affectant deux couleurs différentes et en classant ces fiches dans deux boîtiers différents, de grouper les paramètres descriptifs par sous-classes, éventuellement d'en faire prédominer un ou deux qui pèsent davantage *dans le choix des parties*, etc. Si elle est très astucieuse, la secrétaire de l'agence, sachant qu'elle aura souvent à compulser son fichier, et voulant réduire le temps que lui demandera chaque intervention, inventera en outre des ***signes ou symboles*** pour désigner les diverses valeurs que peuvent prendre les paramètres. Ce seront des sortes de signes sténographiques, abréviatifs, qu'elle pourra inscrire et lire plus vite que des mots, lesquels peuvent former des suites assez longues pour les paramètres qui ne se laissent pas appréhender par des valeurs numériques simples.

Par exemple, pour le paramètre « allure physique », elle mettra les signes ⬭ et ⬯ à la place des mots « assez grassouillet » et « svelte, élancé »; ou encore, pour le paramètre « revenus », elle emploiera les signes abrégés $ et $ à la place des mots « assez fortuné » et « passablement désargenté ». « très fortuné » et « tout à fait à sec » pourront se noter $ et $. Si l'agence traite des milliers de candidatures, elle pourra même se payer une machine à écrire fabriquée de telle manière que, lorsqu'une dactylo tape les mots « svelte, élancé », le signe ⬯ s'imprime automatiquement sur la fiche en face du titre « allure physique » qui désigne le paramètre auquel ce qualificatif se rapporte.

Tous ces arrangements d'écriture font partie de ce qu'on appelle l' « **organisation** » de l'information, et nous y reviendrons plus en détail. Pour l'instant, demandons-nous ce que nous avons fait d'essentiel quand nous avons rassemblé l'information disponible dans un boîtier de façon que nous puissions la consulter et la renouveler facilement, quand ça nous chante. Nous avons exécuté deux choses fondamentales : d'une part, nous avons **codé** l'information, d'autre part, nous l'avons mise en **mémoire.** Voilà deux idées centrales qui vont dominer toute la suite. Examinons-les à tour de rôle.

L'arme la plus puissante de l'homme : le « codage »

On appelle **codage,** en informatique, la représentation des choses par des ***signes*** (*ou* ***symboles***) *selon une* ***convention*** déterminée qui permettra à quiconque la connaît, voyant le signe, d'identifier la chose. En ce sens le langage oral est déjà un codage, puisque, par exemple, l'objet est représenté par le son t-a-b-l-e (pensez s'il vous plaît pour l'instant au ***son*** que vous émettrez en prononçant ce mot, et **non** à sa *forme écrite*). Le signe codé, à cette phase, est purement **oral,** mais il possède déjà les trois caractéristiques essentielles du codage, savoir :

— sa forme physique (en l'espèce un son, plus tard des lettres) peut n'avoir aucun rapport avec l'objet qu'il désigne, mais cette forme s'adapte bien à la gamme des moyens que possèdent les utilisateurs, c'est-à-dire qu'elle

peut être facilement émise et interprétée par eux (on dit aussi « codée » et « décodée »);

— chaque signe « représente » un objet (ou un concept) bien déterminé en vertu d'une règle de correspondance bien convenue; par exemple, il est entendu que le son « table » représente l'objet et le son « chaise » l'objet ;

— cette convention est connue de tout un groupe d'utilisateurs, de façon que n'importe quel membre du groupe, percevant le signe, sait de quelle chose (ou de quel concept) il s'agit.

Dans une phase seconde, les signes *oraux* sont eux-mêmes *surcodés* en signes *scripturaux*, selon le procédé que vous connaissez bien des lettres d'un alphabet phonétique, qui permet de transformer les mots *prononcés* en mots *écrits*. Dans une troisième phase, encore plus élaborée, celle de la secrétaire astucieuse de l'agence matrimoniale, les mots écrits sont codés à leur tour sous forme de petits graphismes qui sont intelligibles — c'est-à-dire « *décodables* » — par tous ceux que la secrétaire a mis dans la confidence de ses conventions : le graphisme veut dire « grassouillet » (qui, à son tour, veut dire ,) et le graphisme veut dire « élancé » (qui, à son tour, veut dire). Remarquez que les dessins *figuratifs* ne forment ***pas*** tout à fait un codage parce qu'ils ont, eux, un rapport physique avec la chose signifiée et prétendent rappeler celle-ci par la simple allure du dessin, ***sans*** une convention spéciale de traduction. Un exemple bien connu où on code graphiquement des *concepts* et non plus des *objets*, est celui des *nombres*, qu'on représente comme vous savez par les fameux graphismes 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, qui constituent les 10 chiffres du système de numération décimale, puis par leurs groupements tels que 10, 25, 138, 2 475, etc. (Mais on aurait pu employer d'autres graphismes.)

Dès sa forme de langue parlée, le *codage* a joué un rôle capital dans le développement de l'humanité. Les anthropologues et généticiens sont unanimes à considérer aujourd'hui que c'est par la découverte de ces modulations de la gorge, discernables à distance, et douées chacune d'un sens connu de tous, bien plus que par l'apposition du pouce et la stature debout, que l'homme s'est le plus nettement détaché de l'état animal. Ce *codage oral* des objets, des actes, des sentiments, confère en effet à ceux qui le pratiquent la formidable puissance de la **communication.** Ainsi les membres d'un groupe qui disposent du langage parlé peuvent se répartir en pelotons au cours d'une chasse pour encadrer le gibier et se communiquer par la voix leurs observations et leurs instructions; grâce à quoi les petits hommes purent attraper de grands éléphants et les domestiquer. Il y a plus. Désormais, l'expérience accumulée par chacun ne reste plus enfouie chez lui seul, elle est transférable à autrui, à celui du moins qui « parle et entend la même langue », c'est-à-dire qui pratique les mêmes conventions pour coder les choses (et concepts) en signes vocaux.

Derrière le codage gît un ressort encore plus profond et plus fondamental de l'aventure humaine, qu'on pourrait appeler **la distillation par l'esprit des principes des choses** (le mot « principes » étant pris ici au sens de la médecine médiévale). Notre organisme nous permet de percevoir et sentir le réel de diverses manières, mais non de recréer les sensations en entier. Par contre, et à la différence des autres animaux, nous gardons assez longtemps dans la mémoire certains *reflets* du réel. Et en liaison avec cette faculté, ainsi sans doute qu'avec les tendances finalistes de nos gènes (que les biologistes appellent « téléonomiques »), nous formons des *projets* où ce réel, qui nous échappe sous sa forme brute, se voit assigner un rôle ordonné : il faut que je passe de l'autre côté de ce ruisseau tumultueux. Comment ces diverses limitations et sollicitations de notre être jouent-elles à notre service ?

Le mécanisme fondamental consiste à ne retenir du réel que les traits *expurgés* qui sont utiles au projet et peuvent entrer dans notre mémoire sans que nous perdions l'information *pertinente*. Nous substituons donc au réel brut ceux de ses éléments qui nous importent, nous les symbolisons par des signes bien mémorisables, et c'est sur ces signes que nous allons ensuite travailler pour retourner vers le réel avec un plan d'action. Dans ce processus, le moment essentiel est de *reconnaître* et *dégager*, dans la bigarrure qui captive nos sens mais étourdit notre pensée, ces éléments *abstraits* — l'eau de la rivière en tant qu'espace à franchir, les rives en tant que points d'appui, la solidité et le poids d'une branche — qui éliminent les 99 % du tableau sensible mais contiennent, en résumé, les filtrats de ces tableaux qui permettront de faire un pont.

Reconnaître dans le réel des composantes simples, constantes et symbolisables, est l'alpha et l'oméga de la science, et c'est aussi l'alpha et l'oméga du codage. On peut dire en bref que le codage permet d'emporter dans notre esprit la quintessence d'une réalité que nous ne pouvons pas conserver vivante et complexe dans la sensation. C'est ainsi que nos limites se transforment en puissances. La conquête du cosmos a commencé le jour où un homme a dit « lune » au lieu de regarder l'astre.

Vous noterez que le mot **code,** ici, est pris seulement dans le sens « message codé » et pas du tout dans le sens « Code Civil ». Il exprime une manière de représenter ce dont on parle et ***non*** un ensemble de règles d'action.

Du cri à la mémoire

Mais la supériorité écrasante que le « codage » oral et transmissible donnait à l'homme sur les autres espèces animales ne suffisait pas encore. Les signes oraux sont volages. Le temps les estompe, le souci les efface. Et ils souffrent de la grave limitation d'exiger que l'émetteur et le récepteur du signe codé soient physiquement en présence l'un de l'autre. Vous n'ignorez pas comment ces faiblesses du langage parlé furent finalement surmontées par l'invention de l'*écriture*, dont tous les livres de classe enseignent avec raison qu'elle marqua l'orée de la civilisation. De notre point de vue, l'important est qu'avec le codage *écrit*, alphabétique, on peut passer d'un enregistrement *fugace* de l'information dans la seule mémoire *biologique* de l'individu, à un enregistrement *permanent* sur un support inaltérable tel que le silex, le papyrus, le papier. Support non seulement inaltérable, mais encore objectif, c'est-à-dire tel que l'information qui s'y trouve inscrite peut être consultée par autrui sans que le sujet

qui lui a donné naissance soit présent, mais à condition bien sûr que ce tiers connaisse les conventions qui ont été employées pour traduire telle chose (ou concept) en tel graphisme.

Voilà apparu le troisième grand pilier de notre affaire : la **mémoire.** En inscrivant l'information disponible sur le demandeur et la recherchée d'un mariage dans les fiches *D* ou *R*, l'agence a mis en œuvre non seulement le grand principe du *codage*, mais encore celui du *stockage dans une* ***mémoire***, *inaltérable*, *consultable*, *renouvelable* et *objective*, qui est en l'espèce : le **fichier.** Notez s'il vous plaît les adjectifs. Si vous inscriviez votre information codée sur un support qui s'évaporerait dans les deux jours, on ne dirait pas que vous l'avez réellement stockée en mémoire. On ne le dirait pas non plus si votre support était inaltérable mais non consultable, comme une feuille de papier qu'après avoir couverte d'écritures vous enfouiriez sous la terre. On ne le dirait pas davantage si votre support était inaltérable et consultable, mais non renouvelable, comme une feuille de papier entièrement couverte et que vous laisseriez à la vue, mais inaccessible derrière une grille. Enfin, on n'appellerait pas **mémoire** un magasin d'informations qui, même s'il était inaltérable, consultable et renouvelable, serait subjectif et non objectif, comme dans le cas où vous instruiriez un homme à apprendre les informations qu'on veut stocker et à les réciter perpétuellement devant un microphone selon son humeur, sans être sûr qu'il recommencera toujours la même chose.

Le prototype le plus simple, le plus efficace et aussi le plus universel et le plus important d'une mémoire satisfaisant ces quatre conditions d'être *inaltérable*, *consultable*, *renouvelable* et *objective*, est le papier blanc recevant des inscriptions durables (mais effaçables), par exemple au crayon. Vous exécutez dix fois par jour des mises en mémoire d'informations sous cette forme, sans réfléchir à l'extraordinaire importance et fécondité de votre acte. Par exemple, quand vous faites une opération et que vous « posez » des résultats intermédiaires, vous les griffonnez rapidement sur le coin d'une feuille; et, sans ce petit geste, toute votre entreprise serait impossible.

Comme vous le reconnaîtrez de plus en plus fortement à mesure que vous avancerez, l'**informatique** des ordinateurs ne fait que continuer et accentuer, avec des moyens plus puissants, le recours ancestral à ces deux méthodes fondamentales de la vie réfléchie et organisée : le **codage** des informations et leur mise en **mémoire.** On croit souvent qu'une révolution consiste à *détruire* les façons de vivre anciennes. C'est une erreur. Elle renouvelle seulement leur sève avec des sucs nouveaux. Une révolution vraiment féconde ne tarit pas, elle élargit les sillages qui ont fait leurs preuves. Si elle doit détruire quelque chose, ce n'est pas la route le long de laquelle les hommes ont progressé, ce sont seulement les obstacles que peut y avoir dressés l'encroûtement des coutumes. La révolution de l'informatique respecte ces grandes lois du développement humain. Nous allons voir qu'elle reprend aussi dans un cadre neuf un troisième sillage ancestral de la vie organisée et réfléchie : celui des **opérations.**

On traite l'information

Retournons à notre agence matrimoniale. C'était très bien d'avoir codé les informations sur le demandeur *D* et la recherchée *R*, et de les avoir mises dans cette mémoire que constitue le fichier. Mais ce n'était pas assez. Ce n'étaient, en un sens, que des

préparatifs pour l'action. Celle-ci arrive maintenant. Qu'allons-nous faire pour donner satisfaction au demandeur?

Pour pouvoir opérer vite, nous avons disposé notre stock d'informations dans ***quatre*** fichiers, et non pas deux. Le fichier *DH* contient les descriptions des demandeurs hommes, numérotés 1, 2, 3, ... A chaque numéro d'une fiche *DH* correspond dans le fichier *RH* (recherchées par les hommes) une fiche portant, sous le même numéro, la description de l'épouse dont rêve le demandeur *DH* ayant ce numéro. Ces deux fiches, *RH* N° tant et *DH* même numéro, ont été créées lors de l'interview avec le demandeur. Deux autres fichiers, *DF* et *RF*, contiennent, sous les numéros qui se correspondent, les descriptions des demanderesses femmes et des hommes recherchés par ces femmes.

Supposons qu'on veuille satisfaire le demandeur homme N° 7, lequel se trouve donc décrit sur la fiche *DH*-7. Dans le fichier *RH*, sous le N° 7, je trouve son idéal de femme. Je sors cette fiche *RH*-7, et, la tenant à la vue, je parcours le fichier *DF* où reposent toutes les demanderesses femmes, faisant défiler les fiches *DF* du doigt tout en les lisant rapidement, m'arrêtant sur celles qui semblent correspondre à la description contenue dans *RH*-7, et extrayant celles qui, après vérification, concordent tout à fait.

▶▶ — Et s'il n'y en a aucune qui concorde tout à fait?

▶ — On écrit à Monsieur 7 qu'on ne peut pas le satisfaire. Supposons, au contraire, que les souhaits enregistrés dans la fiche *RH*-7 concordent avec les fiches *DF*-14, *DF*-25, *DF*-73 et *DF*-108. Cela veut dire que les dames demanderesses N° 14, 25, 73 et 108 satisfont au portrait idéal que m'avait indiqué le monsieur demandeur N° 7. Ce sont donc des mariées *possibles*. Mais notre affaire n'en est pas résolue pour autant. Que faut-il encore vérifier?

▶▶ — Il faut s'assurer que les dames 14, 25, 73 et 108 veulent bien de Monsieur 7 comme mari.

▶ — Fort juste.

▶▶ — Qu'est-ce que je fais pour cela?

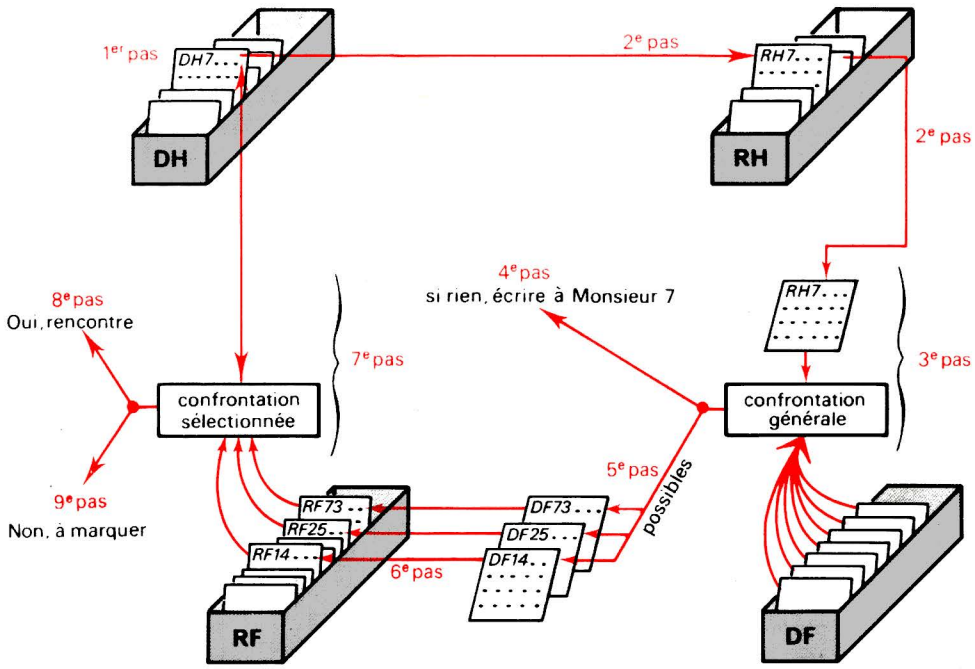
▶ — Vous tirez du fichier *RF* des hommes recherchés par les femmes les fiches portant les numéros 14, 25, 73 et 108 que vous venez d'extraire du fichier *DF*. Elles contiennent les portraits des princes charmants que souhaitent les dames 14, 25, 73 et 108. Et vous comparez ce portrait à la fiche 7 du fichier *DH*, qui contient la description de Monsieur 7. Si l'une ou plusieurs des fiches *RF*-14-25-73 et 108 concordent avec la fiche *DH*-7, vous convoquerez la ou les dames correspondantes pour rencontrer Monsieur 7.

▶▶ — Et sinon?

▶ — Vous ferez aussi bien de noter sur les fiches *DF*-14-25-73 et 108 qu'il ne faudra plus les sortir si c'est la demande de Monsieur 7 qu'on traite.

Voilà donc l'énoncé complet de la procédure que nous appliquons pour mettre en rapport, par l'intermédiaire de nos fichiers, deux êtres qui veulent se marier et peuvent répondre à leurs vœux mutuels.

Cette procédure est représentée sur la figure ci-après et s'appelle, en langage informatique : le **traitement.**



Si je veux confier le traitement à ma secrétaire, je lui détaille la procédure à suivre sous la forme d'une série d'instructions pas à pas, qu'elle devra accomplir exactement et dans l'ordre prescrit :

1er pas : extraire de *DH* la fiche du monsieur à servir;

2e pas : extraire de *RH* la fiche portant le même numéro;

3e pas : confronter à la fiche ainsi extraite de *RH* toutes les fiches de *DF*;

4e pas : si aucune concordance ne se révèle, avertir le monsieur qu'on ne peut pas le satisfaire;

5e pas : si certaines des fiches de *DF* concordent avec la fiche-patron de *RH*, les sortir et les mettre de côté;

6e pas : extraire de *RF* les fiches portant les mêmes numéros que les fiches mises de côté au pas précédent;

7e pas : confronter aux fiches *RF* extraites au pas précédent, la fiche du monsieur à servir, celle qui fut extraite de *DH* au premier pas;

8e pas : s'il y a une ou plusieurs concordances, sortir les fiches pour organiser des rencontres;

9e pas : s'il n'y a aucune concordance, marquer les fiches extraites au 5e pas pour ne plus les soumettre à la confrontation du 3e pas chaque fois qu'on traitera du même monsieur.

Au cœur de tout : le « programme »

En langage informatique, une suite d'instructions de ce genre, prescrivant à chaque pas un acte élémentaire à exécuter, s'appelle un **programme.** Attention à ne pas prendre ce mot dans les autres sens où l'emploie la langue courante, comme par exemple : « programme de la TV », « programme des réjouissances », « c'est tout un programme », etc. En informatique, le mot **programme** n'a qu'une signification et une seule, parfaitement claire et précise.

▶ — Voulez-vous tenter de la définir?

▶▶ — Le programme est l'énoncé, sous forme d'une suite pas à pas, des actes de traitement élémentaires qu'il faut accomplir à chaque pas sur tels ou tels éléments de l'information codée.

▶ — Il existe un mot plus savant que **programme** et qui possède à peu près le même sens, c'est le mot **algorithme** (*). Il désigne lui aussi une suite bien ordonnée d'instructions prescrivant chacune un acte élémentaire à accomplir; mais, jusqu'à tout récemment, il n'a été utilisé que lorsque ces actes portent sur des *nombres ou des grandeurs matérielles.* Le prototype des *algorithmes* est celui que vous appliquez pour faire une division, d'une façon devenue si machinale que vous ne vous apercevez même pas que vous suivez une suite ordonnée d'instructions. Cependant, cette suite est là, sous-jacente. Écoutez-la :

1er pas : isoler dans le dividende une première tranche dont on peut s'apercevoir à vue qu'elle comprend entre 1 et 9 fois le diviseur;

2e pas : trouver par essais mentaux combien exactement de fois, entre 1 et 9, le diviseur est contenu dans cette tranche, et inscrire ce nombre exact de fois comme premier chiffre du quotient;

3e pas : multiplier le diviseur par ce premier chiffre du quotient et soustraire le produit de la première tranche du dividende pour former un premier reste partiel;

4e pas : abaisser le chiffre suivant du dividende;

5e pas : si le premier reste partiel ainsi complété ne contient pas le diviseur, inscrire 0 comme second chiffre du quotient, abaisser un nouveau chiffre du dividende et recommencer comme au 2e pas;

6e pas : si le premier reste partiel complété contient entre 1 et 9 fois le diviseur, recommencer comme au 2e pas.

Etc.

Je vous épargne les détails sur la manière de placer la virgule ou de terminer le cycle, car vous savez tout cela, et je n'évoquais la division que comme exemple d'un *algorithme* que vous exécutiez sans le savoir. On ne sort pas beaucoup de cette notion d'*algorithme* si on l'étend à des cas où les actes élémentaires prescrits ne sont plus

(*) Voir, dans cette collection, le volume « Comprendre les mathématiques ».

seulement des opérations *numériques*. Un algorithme ainsi généralisé pourra comprendre des actes tels que les suivants : extraire une fiche, introduire une fiche, superposer deux graphismes ou deux tableaux de marques pour voir s'ils coïncident, assembler plusieurs fiches dans un groupe, coder une information, la numéroter. Dès lors que de tels actes élémentaires sont clairement définis et facilement exécutables — par l'esclave — dans un ordre parfaitement prescrit, leur succession ordonnée constituera un algorithme. Cependant, si la notion d'algorithme s'étend sans peine aux cas où les opérations élémentaires ne sont pas nécessairement numériques, on n'aime guère l'employer lorsque leur suite est *trop ramifiée*. C'est alors qu'on la remplace par la notion de **programme.** Par exemple, on hésitera à dire que pour établir les 350 000 feuilles de paie des employés de la SNCF, à partir de leur identité et de leurs fonctions, on met en œuvre un algorithme. On doit en effet accomplir plusieurs centaines d'actes élémentaires se groupant en de nombreux cycles qui se conditionnent et s'enlacent les uns les autres. C'est pourquoi on parlera, de préférence dans ce cas, de programme. On tend donc à réserver le mot d'*algorithme* à des suites relativement courtes d'instructions à exécuter l'une après l'autre pour parvenir à un résultat final pas trop distant des données initiales, de façon que le passage des données initiales au résultat final puisse être embrassé du regard et coulé dans un concept unique. Lorsque les suites d'instructions sont d'un ordre de grandeur plus complexe, on emploiera plutôt le mot de **programme.** *En somme, un* ***programme*** pourra contenir plusieurs algorithmes.

Les agences matrimoniales nous en donnent un bon exemple. En effet, les procédures qu'elles doivent mettre en œuvre pour découvrir dans leurs fichiers des couples assortis sont plus compliquées que nous ne l'avons dit. Songeons, par exemple, que la confrontation entre un portrait désiré et les portraits réels disponibles ne peut jamais aboutir à une concordance *parfaite*. Il faut donc allouer à chaque trait idéal recherché par le demandeur une certaine *marge*. Et ces marges se conditionnent les unes les autres. Par exemple, le demandeur dira : « Je la désire blonde, mais j'accepterais une rousse si elle est assez riche. » Vous voyez tout de suite que ces *desiderata*, plus réalistes et plus souples, exigent une recherche plus subtile. Il faudra instruire la secrétaire que, quand son œil rencontre la mention « rousse », elle ne devra plus rejeter la fiche correspondante, mais consulter aussitôt du regard la valeur inscrite en face du paramètre « fortune », et ne décider le rejet que si cette valeur se trouve inférieure à un seuil fixé. Vous voyez que la seule extraction des fiches répondant à une commande *avec alternative*, telle que « blonde de toute fortune *ou* rousse possédant au-dessus de telle somme », exige qu'on mette sur pied un *algorithme* en soi. Comme il y aura de nombreux autres cas de ce genre avec des marges et des alternatives se conditionnant les unes les autres, l'appairage de candidats qui se conviennent finira par exiger un vrai *programme* pour le dépouillement des fichiers, un programme assez vaste et délicat, comportant de nombreux *algorithmes partiels* pour confronter deux fiches sur la base de critères doués de marges et liés entre eux.

Vous sentez bien que, quelque raffinés et interdépendants que soient les désirs des demandeurs, on pourra toujours écrire une série d'instructions pas à pas dont l'exécution aveugle mais rigoureuse assurera que ces désirs seront exactement satisfaits, pour autant qu'ils peuvent l'être avec les personnes fichées. On pourra même assurer qu'en suivant scrupuleusement le programme de dépouillement des fichiers, on ne manquera aucune des possibilités potentielles d'appairage qui y sont enfouies.

Voici ce que fait l'ordinateur

Nous reviendrons bien sûr sur cette vision de programmes longs et complexes. Pour l'instant, récapitulons les premiers aperçus que nous avons glanés sur le fondement du métier de marieur. Nous avons vu qu'en gros il consiste à faire les trois choses suivantes :

— **coder** l'information significative (dans le cas présent les valeurs des paramètres décrivant les candidats);

— mettre cette information codée dans une **mémoire** convenablement organisée;

— **traiter** les données qui s'y trouvent ainsi inscrites dans la mémoire selon un **programme** qui aboutit au but recherché (dans le cas présent, ce but est de dégager les couples dont les *desiderata* s'accordent).

Eh bien, ami lecteur, dans ces trois activités du bon marieur, réside plus généralement l'essence de toute l'informatique. Comme on a pris l'habitude (un peu rapide, vous le verrez) d'appeler **donnée** tout élément d'information lorsqu'il a été codé selon une certaine convention, comme le mot ***donnée*** *se dit en anglais* ***data***, et que le mot **traitement** se dit dans la même langue **processing** (qui signifie littéralement : soumettre à un processus), ce que nous appelons en français l'**informatique** s'appelle en anglais **data processing.** Les grands ordinateurs ne font rien d'autre que d'assurer le codage des informations, la mise en mémoire des données qui en émanent, leur traitement programmé, et, au bout, la présentation du résultat. Le tout ressemble beaucoup à ce que nous aurions fait de main d'homme, mais les ordinateurs exécutent ces travaux infiniment plus vite.

Que veut-on dire quand on dit qu'une machine « comprend » une information?

En réalité, il n'y a pas que par la *vitesse* que le traitement d'informations codées en « **ordino** » diffère du traitement en « **homo** ». La *forme* physique des informations codées est également différente dans les deux cas. Seules sont les mêmes : la structure logique des programmes, la signification des données, la nature des résultats. Par contre, les données elles-mêmes circulent dans le monde « **ordino** » non seulement beaucoup plus vite, mais encore sous une toute autre incarnation que dans le monde « **homo** ». Cela tient à ce que l'organisation sensorielle de l'ordinateur ne ressemble pas du tout à celle de l'homme. L'ordinateur est fait de circuits électriques et magnétiques auxquels, jusqu'ici, on n'a pas réussi à faire comprendre des signes graphiques tels que ⊗, $\mathcal{S}$, $\mathcal{S}$, etc., que l'œil et le cerveau humains arrivent au contraire fort bien à saisir et interpréter d'un seul coup.

▶▶ — Je vous demande pardon de vous interrompre, mais qu'est-ce que c'est exactement, pour une machine, que « comprendre » un signe?

▶ — Vous avez raison de poser la question. Peut-être est-il un peu hâtif d'employer ce mot « comprendre » aussi bien dans le monde « **ordino** » que dans le monde

« **homo** ». Peut-être la « compréhension *homo* » est-elle capable de nuances, de profondeurs, d'échos que la « compréhension *ordino* » n'atteindra jamais. Nous discuterons ce problème dans le dernier chapitre. En attendant, il sera légitime d'employer le mot « comprendre », aussi bien dans l'univers *ordino* que dans l'univers *homo*, si on précise ce qu'on entend par là. Disons donc que « comprendre » un signe impliquera que l'organisme qui « comprend » réagit à la présentation du signe par la formation de « quelque chose » — un influx nerveux, un courant électrique, un contact mécanique — d'une manière qui est à la fois :

— **spécifique,** c'est-à-dire que les réactions à deux signes différents sont différentes (« spécifique » ici s'oppose à ambigu ou indiscriminé);

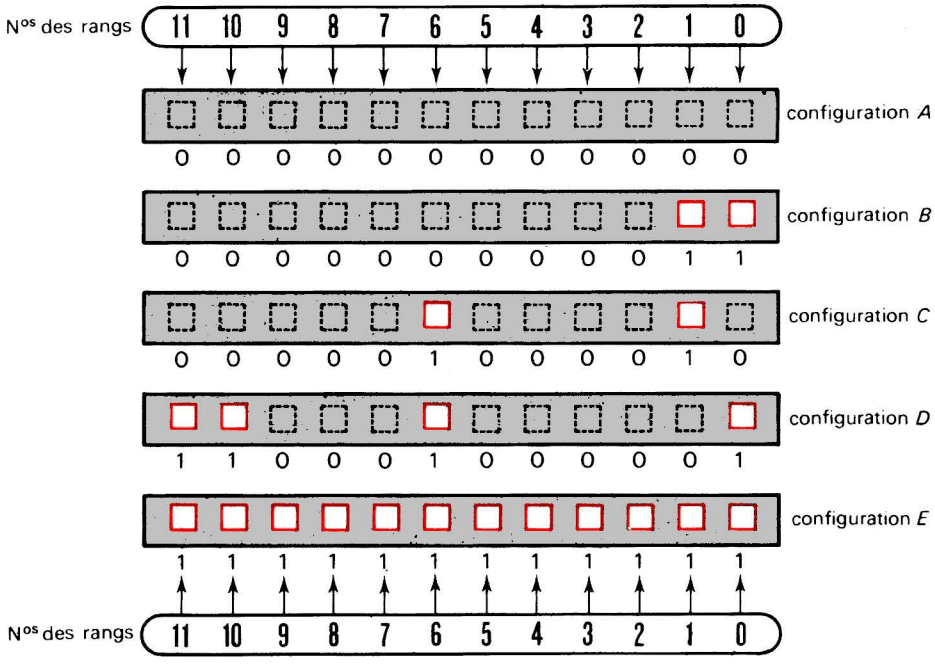
— **utilisable,** c'est-à-dire que ce « quelque chose » peut à son tour, seul ou en combinaison, provoquer des chaînes de réactions désirées, (« utilisable » ici s'oppose à stérile ou rétif);

— **contrôlable,** c'est-à-dire qu'on peut vérifier si les réactions sont bien conformes à quelque « patron » qu'on s'était fixé (« contrôlable » ici est synonyme d'objectif, ou réglé, et s'oppose à fantaisiste ou arbitraire).

Vous comprendrez mieux — eh oui, comprendre à la manière *homo* — cette acception du mot « comprendre » applicable aussi bien en *ordino* qu'en *homo*, si vous réfléchissez au type de codage qu'un ordinateur « comprend ». Supposez que je dispose d'un carton sur lequel sont marqués, en pointillé, le long d'une ligne, un certain nombre — mettons 12 — de petits carrés ou cases discernables par leur place (ou « rang ») dans la filière. Je peux perforer l'une ou l'autre ou plusieurs de ces petites cases, ou toutes, à mon gré, de manière à les transformer en ***trous***. Les carrés qui resteront non perforés s'appelleront des ***pleins***. Afin de pouvoir noter par devers moi, selon mes habitudes d'*homo*, les perforations qui ont été faites, je décide d'un codage dans lequel chaque *plein* sera représenté par le chiffre 0 et chaque *trou* par le chiffre 1. La filière des 12 cases toutes occupées par des *pleins* sera donc représentée par 12 chiffres 0 côte à côte, et la filière des 12 cases toutes occupées par des *trous* sera représentée par 12 chiffres 1 côte à côte.

Je peux maintenant faire des tas de conventions qui font correspondre à des choses, ou à des concepts, telles ou telles *configurations* de pleins et de trous. Par exemple, je peux convenir que la configuration *B*, où j'ai perforé les cases de rang 0 et 1, et qu'avec mes graphismes *homo* je note 000000000011, signifiera « svelte, élancé ». La configuration *C*, qui se note 000001000010, pourra signifier, mettons, « universitaire ». La configuration $D = 110001000001$ signifiera, mettons, « fortuné », et la configuration *E* entièrement occupée de trous signifiera « juste libéré de prison où il purgeait une peine pour faillite frauduleuse » (voir figure page suivante).

En répartissant de diverses manières, parmi les douze cases allouées, des pleins et des trous, qui donnent à chaque case *deux* « états » possibles, on peut composer $2^{12} = 5\,096$ configurations différentes, ce qui permet déjà de coder pas mal de choses et de concepts. Si on ajoute *une* case, on peut composer le *double* de configurations différentes, le *quadruple* si on ajoute *deux* cases, etc. On peut d'ailleurs concevoir des codages plus subtils. Par exemple, si on a subdivisé les paramètres descriptifs des personnes en quatre catégories (physiques, caractériels, sociaux, culturels) on peut

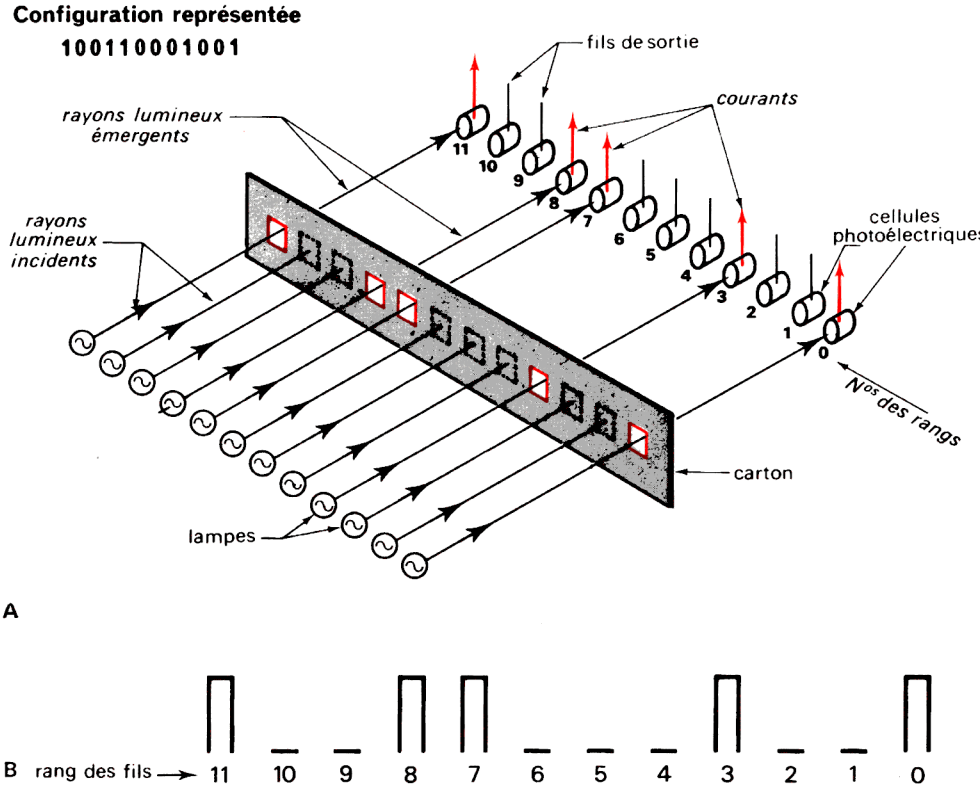


Attention! Les rangs sont numérotés de droite à gauche à la façon dont ils se présentent dans l'écriture des nombres. Pour le nombre 2 493, le chiffre 3, le plus à droite, est de rang 0, puis le chiffre 9, plus à gauche, est de rang 1, puis le chiffre 4, plus à gauche, est de rang 2, etc.

convenir que les deux premières places de la filière serviront à préciser à quelle catégorie doit se rapporter la configuration perforée dans les autres places. Etc.

Devant un carton perforé selon une certaine configuration (ici, sur la figure **A** : 100110001001) pour signifier, dans le code adopté, une certaine chose (ou un concept) *X* j'arrange maintenant une série de 12 lampes qui envoient vers chaque case un rayon lumineux. De l'autre côté du carton sont placées 12 cellules photoélectriques. Vous savez sans doute ce que c'est que cet engin : c'est un petit cristal qui a la propriété, quand il est frappé par de la lumière, d'engendrer un courant électrique dans un circuit où il est inséré. Dès lors, les cellules placées en face des pleins restent inertes parce qu'ils ne reçoivent pas de lumière, tandis que celles placées en face des trous réagissent au message qui leur est présenté en produisant chacune un courant. Ce courant possède partout la même intensité si les intensités des éclairages, la taille des trous, la nature des cellules et celle des circuits sont uniformes.

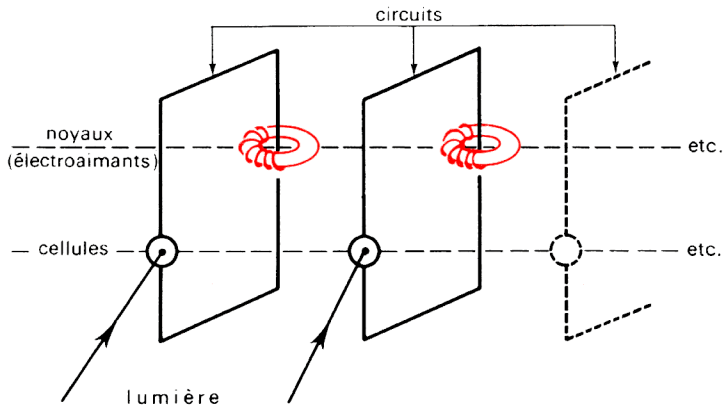
Si on s'arrange pour présenter le carton aux lampes pendant un temps très court (par exemple en le faisant suivre immédiatement par un carton entièrement plein) on parvient finalement à ce que la chose ou le concept *X* soit traduit sous la forme d'une configuration d'*impulsions électriques* dans les 12 fils (voir figure **B**). Tel est le « quelque chose » qui répond aux signes présentés à la machine. Voyons si ce « quelque chose » — ici donc un train d'impulsions ou d'absences d'impulsions



sur 12 fils — est conforme aux trois conditions que nous posions plus haut pour pouvoir dire que la machine a « compris » le message.

• 1re *condition* : la réponse est-elle *spécifique*? Évidemment oui. A toute configuration de trous, donc à tout message, correspond une réaction et une seule de la machine. A deux messages différents correspondent deux réactions différentes.

• 2e *condition* : la réponse est-elle *utilisable*? Si vous étiez ingénieur électronicien, vous répondriez oui tout de suite, car le spectacle des 12 impulsions ou non-impulsions de courants électriques dans 12 circuits ferait immédiatement surgir dans votre esprit mille combinaisons à faire avec ces quantités. N'étant pas ingénieur électronicien, vous devez me faire confiance quand je vous affirme que ces combinaisons sont possibles, et pas difficiles. D'ailleurs, je vous en montrerai quelques-unes dans un chapitre ultérieur. Vous y verrez par exemple comment, avec deux trains d'impulsions différentes occupant deux douzaines de circuits, on peut fabriquer, dans une troisième douzaine de circuits, un train d'impulsions comprenant toutes celles qui comprennent les deux autres, ou seulement celles qu'ils ont en commun, ou des impulsions partout où il y a deux absences d'impulsions, etc. On peut aussi placer, dans les 12 circuits des cellules photoélectriques, 12 petits électroaimants (voir p. 20). Lorsqu'un électro-aimant (convenable) reçoit une impulsion de courant, son noyau s'aimante et garde son magnétisme (jusqu'à ce qu'une impulsion de sens contraire le détruise).



Dès lors la batterie de ces noyaux constitue une *mémoire* où l'information 100110001001 se trouve stockée, et on peut à son tour l'utiliser pour d'autres opérations. Par exemple, ayant inscrit ce message codé dans la mémoire des électroaimants, on peut lui comparer tout autre message, et s'arranger pour que, sur un fil traversant l'ensemble des électroaimants, sorte une impulsion de courant chaque fois que le message présenté est le même que ce « patron ». On peut dire alors que par cette impulsion la machine nous notifie qu'elle a perçu une concordance entre le patron et un autre carton.

• 3e *condition* : la réponse est-elle *contrôlable*? Évidemment oui. On a mille moyens techniques de vérifier qu'une configuration de trous faits dans le carton donne bien naissance à la configuration convenue d'impulsions électriques dans les circuits. Il n'y a rien d'aléatoire dans le processus.

Vous apprendrez à connaître bien d'autres combinaisons utiles auxquelles se prêtent les trains d'impulsions électriques. Elles sont en vérité infiniment moins variées et subtiles que celles qu'effectue, avec les influx nerveux, le système mental de l'homme. Mais elles sont nettement plus variées et subtiles que les combinaisons qu'on peut réaliser en mécanique avec, par exemple, des engrenages et des cames, et surtout elles peuvent se dérouler des millions de fois plus vite que dans le cerveau ou dans les mécanismes. Vous admettrez donc qu'on peut agencer une machine de façon qu'elle effectue, sur les impulsions traduisant les messages codés, tous les traitements qu'appellera un « programme » et qui conduiront à un résultat final exploitable.

J'espère que vous voyez maintenant dans quel sens on peut dire qu'une machine du genre de celle que j'ai esquissée, et qui est un très élémentaire squelette d'ordinateur, « ***comprend*** » les informations qui ont été codées sous la forme de trous dans un carton. Dès lors nous pourrons dire qu'*on se marie par ordinateur exactement comme on se marie par une agence*. Les candidats fournissent à l'ordinateur, sur eux et sur leurs souhaits, les mêmes renseignements qu'ils fournissent à une agence. Le gestionnaire de l'ordinateur *code* ces renseignements sous une forme que l'ordinateur *comprend*, et les inscrit dans sa *mémoire*. Puis, quand on doit répondre à des demandes, l'ordinateur exécute, avec les données qu'il a en mémoire, le traitement prescrit par un *programme*. Ce programme a été conçu par le maître de l'ordinateur et ressemble de très près à celui que nous avons élaboré plus haut quand nous étions une agence

matrimoniale à opérations manuelles. Au terme du traitement prescrit par le programme, l'ordinateur sort, par exemple en tapant leur liste à la machine, toutes les femmes fichées qui concordent avec les désirs d'un homme donné, ou l'inverse. En somme, l'ordinateur remplace la secrétaire qui manipule le fichier. Et avec avantage, car le programme pourra être exécuté en une seconde au lieu de plusieurs heures. Du coup, on pourra s'enhardir à formuler des exigences de plus en plus raffinées, comportant des marges et conditions de plus en plus nombreuses et enchevêtrées. Les programmes nécessaires pour décortiquer et satisfaire de telles demandes complexes exigeraient, en *homo*, des temps prohibitifs, alors qu'en *ordino* ils resteront exécutables dans des temps admissibles. De ce fait, les candidats au mariage pourront être analysés et servis plus finement en *ordino* qu'en *homo*. Disons le mot, on scrutera leur être intime plus à fond et on en tiendra mieux compte. Ainsi, loin de « mécaniser » les mariages, l'ordinateur les « humanisera ». Ce n'est là que la première touche d'une vue que nous ne cesserons d'élargir au cours de ce livre, et surtout dans le dernier chapitre.

L'ordinateur est un esclave

Un dernier mot pour clore ce survol préalable du territoire que nous allons sillonner. J'ai dit au début que nous allions nous marier par ordinateur. C'était, vous l'avez sans doute compris, une boutade pour vous provoquer un peu. En fait cette annonce était trompeuse.

▶ — Voyez-vous maintenant pourquoi?

▶▶ — Parce que l'ordinateur ne marie personne. Tout ce qu'il fait est de dégager, dans une population de candidats au mariage, deux partenaires *A* et *B* dont les souhaits exprimés et les caractéristiques déclarées concordent.

▶ — Bien sûr. Et nous venons de voir qu'il le fait selon des directives exactement semblables à celles qu'emploirait une agence, mais de façon plus complète, plus fine, et considérablement plus rapide. Une fois sortis des fichiers deux candidats *A* et *B* qui se conviennent ***a priori***, *selon leurs propres desiderata*, ces personnages conservent tout leur libre arbitre. Ils se rencontrent, vérifient s'ils se sont correctement décrits, éprouvent s'ils se plaisent, peuvent se quitter, se tester, se compléter... S'ils se marient, ce sont eux qui le font, pas la machine. Et ils le font après avoir pleinement déployé leur registre humain personnel. L'ordinateur leur aura simplement épargné des rencontres dont on pouvait être sûr d'avance qu'elles seraient inutiles. Il aura *sélectionné le champ des possibles*, et, notez-le, non pas selon ses propres humeurs, mais selon les règles mêmes que les candidats lui auront dictées. Dans le cadre de ces règles, l'ordinateur aura assuré une *sélection parfaite*, c'est-à-dire à la fois *exhaustive* et *fidèle*. Si la sélection était imparfaite — soit qu'elle manque des appairages qui auraient été possibles, soit qu'elle propose des appairages qui s'avèrent dès l'abord impossibles — la faute ne peut en incomber qu'aux règles ou aux informations que les hommes ont imparties à l'ordinateur (nous supposons bien sûr que le programme lui-même est correct, c'est-à-dire ne contient pas des erreurs matérielles comme de sauter un échelon ou de mal recopier des données; ces erreurs peuvent toujours être

éliminées par une révision soigneuse et des essais préalables). Nous acquérons ici un autre aperçu important qui prendra une forte gravitation par la suite : l'ordinateur n'est jamais qu'un *outil de service* pour l'homme. C'est une secrétaire plus efficace. (Mais, hélas, moins galbée!)

Articulation de l'ouvrage

Le panorama introductif qui précède nous indique les lignes de force du paysage que nous devons maintenant parcourir.

Puisque vous avez agréé l'idée qu'on peut arranger des machines de façon qu'elles exécutent une grande variété de combinaisons avec des trains d'impulsions électriques, je peux tenir le principe pour acquis, reporter en seconde position la *manière de réaliser* ces combinaisons dans ce qu'on appelle le « **hardware** », et commencer par vous dire comment on les *conçoit*, ce qu'on appelle le « **software** ». Ces deux mots, devenus illustres, empruntés à l'anglais parce qu'ils n'ont pas d'équivalents dans les autres langues et parce que l'informatique est née aux États-Unis, signifient littéralement : marchandise (ware) dure (hard) ou molle (soft). Le mot « hardware » était employé depuis longtemps pour signifier « quincaillerie », et le mot « software » pour signifier « parlote ». Les fondateurs américains de l'informatique y ont recouru figurativement et ironiquement pour désigner, d'une part, l'ensemble des organes électriques et mécaniques qui constituent un ordinateur — le « **hardware** » — et, d'autre part, l'ensemble des ordres qui spécifient ce que le **hardware** doit faire — le « **software** ». Si on se plaçait en thérapeutique appliquée à un malade, le **hardware** serait constitué par les remèdes chimiques ou physiques, et le **software** par le diagnostic et l'ordonnance médicale.

Au début de l'informatique, l'attention se portait exclusivement sur le hardware, merveille toute neuve qui absorbait l'essentiel des talents et des efforts. Les problèmes qu'on posait à l'ordinateur (qui s'appelait calculateur) étaient simples, et chaque utilisateur ajustait pour son compte les programmes nécessaires. Mais la dévorante gourmandise de la civilisation industrielle rendit rapidement les machines et les problèmes gigantesques. De ce fait, la préparation des programmes se compliqua à tel point qu'elle dut devenir un art en soi. C'est alors qu'il fallut la distinguer, par un mot spécial, de la « quincaillerie » qui exécute les ordres. Et comme cette dernière avait été déjà baptisée « hardware », on forgea le nom de « software » pour désigner ce que notre esprit demande au hardware de faire. Le contraste est, en effet, heureux. Songez cependant que s'il est naturel d'appeler « dure » la marchandise matérielle, il y a quelque irrévérence à appeler « molle », et même « marchandise », l'œuvre de notre esprit...

La fameuse compagnie IBM fut la première à comprendre qu'à mesure que les applications des ordinateurs s'amplifiaient et pénétraient toutes les phases de la vie, les organes qui tournaient en dedans, surtout s'ils tournaient bien, perdaient de leur importance aux yeux de la clientèle, par rapport à la manière de s'en servir. De même, le public ne songe plus guère aux pistons qui travaillent sous le capot des voitures, et qui sont l'affaire des spécialistes. Il s'intéresse essentiellement aux commandes et au confort. La compagnie IBM fit sa fortune en vendant les commandes plutôt que les

pistons de l'informatique, c'est-à-dire en soignant d'abord et présentant surtout le *software* dont son *hardware* était capable. De même que 80 % des automobilistes conduisent leur voiture sans savoir comment fonctionne exactement le moteur à essence à quatre temps, de même les usagers modernes des ordinateurs n'ont besoin de connaître que leur software et peuvent ignorer à peu près tout du hardware. A la limite, on peut assimiler le software à la littérature ou à la musique, et le hardware à l'imprimerie ou au tourne-disques.

Je consacrerai donc la 1re partie à l'étude du software, c'est-à-dire essentiellement à l'art du codage et des programmes. La 2e partie traitera quand même du hardware, car si un usager peut à la rigueur l'ignorer, il reste fascinant pour le curieux éclairé que vous êtes, d'abord parce qu'il met en œuvre les plus brillants monuments d'ingéniosité de notre temps, ensuite parce qu'il continue d'évoluer avec un extrême dynamisme et qu'il conditionne les performances ultimes de ces fantastiques usines à traiter les données. Vous trouverez les grandes applications des ordinateurs dans la 3e partie. En finale, nous aborderons les problèmes passionnants que l'informatique a soulevés sur le plan social — nous mène-t-elle à la disparition du travail? Et sur le plan de la destinée humaine — nous mène-t-elle à la robotisation de l'espèce? Par la page que j'ai mise en épigraphe de ce livre, et par sa table des matières, vous savez déjà quelle est ma thèse sur ce point capital : c'est que l'ordinateur, loin d'éteindre en nous le feu de Prométhée, exigera davantage de lui et avivera son éclat.

Dans ce premier survol, on vous a appris ce que c'est que *coder* une information au moyen de symboles oraux ou écrits, et que coder est le propre de l'homme. De quoi la conséquence naturelle est qu'on cherche à enregistrer les expressions codées des thèmes auxquels on s'intéresse dans une *mémoire,* capable de les conserver plus longtemps et plus objectivement que notre souvenir. Si en outre les symboles inscrits dans la mémoire sont faciles à manipuler, il devient possible de soumettre nos informations à des *« traitements »,* c'est-à-dire à des suites d'opérations élémentaires. La succession des *ordres* ou *instructions* qu'on donne à quelqu'un pour lui prescrire d'exécuter à chaque pas une opération élémentaire sur l'information codée s'appelle un *programme,* ou encore un *algorithme* si la succession est relativement courte. Forts de ces notions, vous avez été admis à examiner comment le « quelqu'un » qui exécute un programme peut être une machine, capable de comprendre, d'une part les représentations codées des informations et, d'autre part, les ordres de traitement qu'on lui donne. D'où résulte qu'une telle machine ne sera pour nous qu'un esclave..., ce qui ne diminue pas son énorme valeur dès lors qu'elle sait effectuer les traitements qu'on lui prescrit des milliers ou millions de fois plus vite que nous. Cet esclave supérieurement agile sera l'*ordinateur,* et on a terminé le chapitre en présentant la marche que l'ouvrage suivra pour faire comprendre son fonctionnement. A cette occasion vous avez fait connaissance avec deux mots fameux : *« hardware »* qui désigne les organes mêmes, électroniques et mécaniques, des machines, et *« software »* qui désigne les programmes qu'on leur fait suivre.

Première partie

le software

Le codage

L'art des programmes

Les langages

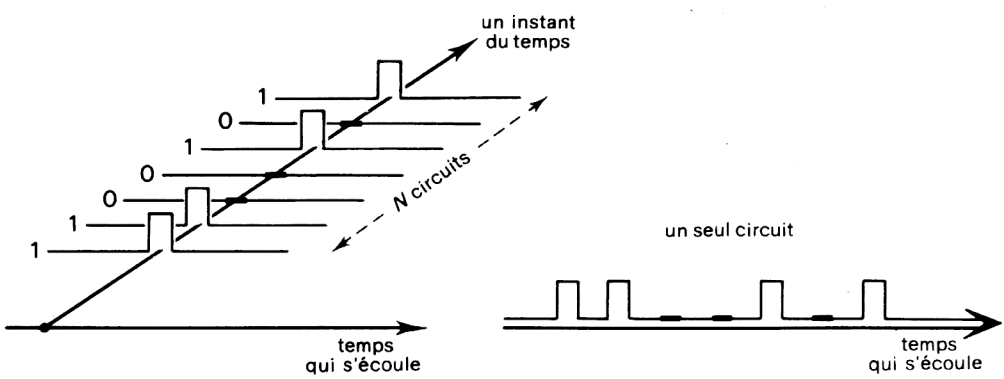
1

Le codage

Le principe des impulsions binaires

Vous avez déjà acquis plusieurs notions fondamentales de l'informatique, dont la première est la suivante : pour qu'une information puisse être comprise et traitée par un ordinateur, on la « code » sous la forme d'un train de N impulsions électriques dans un faisceau de N circuits. Ces impulsions se produisent bien sûr à un même instant sur tous les circuits, et le codage se traduit seulement par le fait qu'il y a une impulsion sur certains circuits et qu'il n'y en a pas sur certains autres. On dit que c'est un codage « parallèle » ou « par configuration instantanée ». Pourquoi « parallèle » ? On pourrait répondre facétieusement : parce qu'il n'est pas « en série ». Expliquons-nous.

Rien n'interdirait, tout conseillerait au contraire de concevoir un codage obéissant aux *mêmes conventions*, mais où les impulsions et les non-impulsions se produiraient, dans un seul et même circuit, à des instants *successifs*, le train complet de N impulsions occupant donc dans le temps N fois la durée élémentaire d'une impulsion, plus les durées des repos entre impulsions. En fait, les codages humains — oral et écrit — sont du type « série » parce que nous ne pouvons émettre ou écrire qu'un son ou un signe à la fois. Des êtres intelligents, mais qui ne sauraient proférer que les deux sons « oui » et « non », pourraient encore exprimer beaucoup d'idées s'ils s'étaient tous entendus sur une convention — un code — permettant d'associer à chaque idée une succession déterminée de ces deux seules émissions sonores. Ainsi, dans un code à douze places,



Codage parallèle **Codage série**

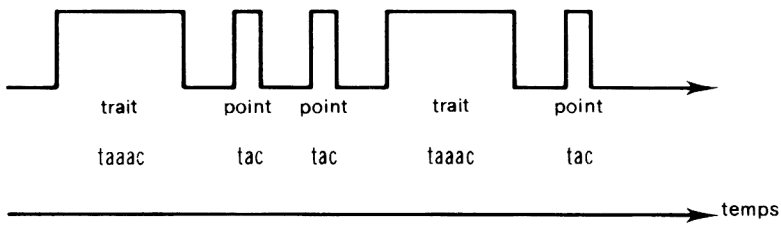
papa se dirait par exemple « non non non non non non non non non non oui », manger se dirait, par exemple, « oui non non non non non non non non non oui », etc. Ce serait un codage « série ». Les premiers ordinateurs, conçus à une époque où le matériel coûtait cher et où les exigences de vitesse n'étaient pas encore poussées à l'extrême, employaient un codage série. Un tel codage en effet consomme du temps, mais pas de circuits. C'est exactement l'inverse avec un codage parallèle, qui permet de situer l'ensemble de l'information élémentaire en un seul instant, mais exige pour cela qu'il y ait autant de circuits marchant en parallèle que de places dans le code. Aujourd'hui, alors qu'on veut aller toujours plus vite, et que le matériel coûte moins cher, la quasi-totalité des ordinateurs sont construits « en parallèle ».

Même en parallèle, un codage qui ne se sert que de **deux** états discernables (impulsion ou non impulsion, 0 et 1, non et oui) et que pour cette raison on appelle « **binaire** », paraît bien primitif si on le compare à l'extraordinaire souplesse et subtilité qu'on atteint avec les quelques 35 sons élémentaires distincts que notre gorge peut émettre et notre oreille identifier, et encore plus si on le compare avec l'infinité des signes graphiques que notre plume peut tracer et notre œil reconnaître. C'est incontestable. Mais en contrepartie de son infantilisme sur le plan logique, le codage « binaire » présente plusieurs vertus techniques très appréciables.

La première est son extrême simplicité. Les organes sensoriels et intellectuels de l'homme ont une complexité, un registre, une finesse, une aptitude à différentier et intégrer les formes qu'aucun organe physique créé par l'homme n'a encore approché, ni de très loin. Lorsqu'on veut produire en masse, de façon qu'ils soient bien identiques, contrôlables et bon marché, des organes physiques qui sachent présenter et discerner *divers états*, on s'aperçoit que la meilleure solution est de restreindre ces états à *deux*, qui diffèrent de façon très tranchée, genre « tout » ou « rien » : trous et pleins dans un carton, denture ou pas denture dans un engrenage, contact ouvert ou fermé dans un relais, noyau aimanté ou pas aimanté, circuit électrique excité par une impulsion ou restant neutre. De tels dispositifs *binaires sont plus sûrs* que les modulations graduées de la voix, c'est-à-dire qu'on a moins de chances de confondre les signes. C'est ce qu'avait déjà compris Chappe lorsqu'il créa le premier télégraphe avec d'énormes bras mécaniques qu'on baissait ou levait sur les sommets du paysage de façon qu'ils soient visibles de loin. Il s'était arrangé pour que tout message pût se ramener à une suite de bras levés ou bras baissés, et pour cela, chaque lettre de l'alphabet avait été codée sous forme d'une succession déterminée de ces deux positions. Comme on peut réaliser 2^N configurations binaires quand on offre au message codé N places, il suffisait de prendre $N = 5$, c'est-à-dire qu'avec 5 suites de bras levés ou baissés on peut réaliser 32 combinaisons, donc plus qu'il n'en faut pour accommoder notre alphabet de 26 lettres. Le télégramme Chappe utilisait donc un codage binaire série.

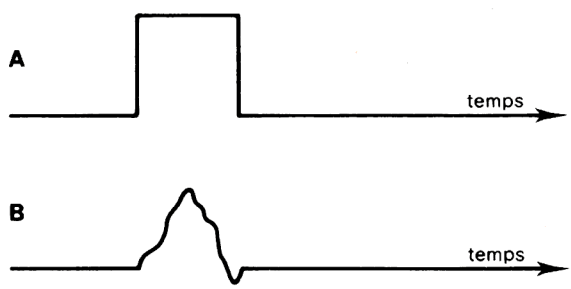
Plus tard, lorsque les signaux optiques du télégraphe Chappe, qui étaient lus et transmis à la vue, furent remplacés par des signaux électriques voyageant sur des fils, Morse adapta les idées de Chappe dans un code qui porte toujours son nom et qu'on emploie toujours. Morse utilise déjà comme signaux des impulsions de courant d'une amplitude uniforme et n'introduit, lui aussi, que deux états. Mais ces deux états ne sont pas tout à fait le « oui » et le « non » des ordinateurs. Morse les définissait par deux *durées d'existence* nettement distinctes, l'impulsion de durée longue s'appelant un « trait » et celle de durée courte un « point ». Ce principe de codage binaire est

Le code Morse d'une lettre



employé en télégraphie parce que les signaux traversent de grandes distances dans des milieux variés et passent par une multiplicité de relais. Ils peuvent de ce fait être retardés de façon incontrôlable, ce qui rend aléatoire leur instant d'apparition et ne permet donc pas de compter sur la « place » d'une impulsion pour caractériser un message. Cette difficulté ne se présente pas dans le sein d'un ordinateur, où on peut contrôler très exactement les instants auxquels les impulsions sont admises à apparaître — ou ne pas apparaître. Cela devient d'ailleurs possible aussi, grâce aux nouvelles techniques de synchronisation, sur les longs circuits télégraphiques, et c'est pourquoi le vieux code Morse tend à disparaître au profit du code binaire pur, où toutes les impulsions ont la même amplitude et la même durée et ne prennent un sens que par leur présence ou absence à des instants déterminés ou sur des circuits déterminés.

Ces impulsions standard ont une autre vertu que la simplicité, c'est la **sûreté.** Songeons un moment au côté pratique des choses. Vous n'avez pas besoin d'être électronicien pour vous rendre compte qu'une impulsion électrique ne traverse pas des circuits compliqués sans s'abîmer quelque peu. D'autant qu'elle prétend avoir des flancs verticaux, c'est-à-dire que, dans le circuit qu'elle traverse, le courant devrait, idéalement, passer de zéro à la pleine amplitude en un temps *nul.* Or vous n'ignorez pas que l'inertie règne dans la nature autant que chez les hommes, et donne à celle-ci « l'horreur de l'instantanéité ». En fait, tout établissement, ou toute extinction, de courant prend un certain temps, non nul, et plus on veut forcer ce temps à être bref, plus le circuit rechigne et fait « trembler » le courant. De tout cela résulte qu'après avoir un peu travaillé et voyagé, une impulsion de courant n'a plus la belle allure de la figure **A,** mais l'allure quelque peu fatiguée et cabossée de la figure **B.** De même un trou dans un carton, si on passe dessus ou dedans un certain nombre de fois, prend des bords effilochés. Cependant, si on alloue un espace suffisant de carton avant la



prochaine case, il suffira que les effilochures ne s'étendent pas au delà de cet espace, pour qu'on puisse reconnaître avec certitude qu'il y a un trou en la case considérée. Dès lors, vous voyez clairement tout ce qu'on gagne en arrangeant l'*ordino* de façon que la seule chose à savoir pour traiter pertinemment l'information est s'il y a « trou » ou « pas trou ». Le bénéfice fondamental de ce mode « tout ou rien », c'est que les déformations subies par l'impulsion au cours de son labeur *n'empêchent pas* le traitement de rester toujours correct. Il n'en irait pas ainsi si l'information nichait dans le contour même du trou ou du courant, dans ses gradations, ses méandres ou sa taille. Dans ce cas, toute déformation impliquerait que l'information a changé.

En *ordino*, on s'arrange pour laisser, entre deux trains d'impulsions, un temps creux égal à la durée même d'une impulsion, ce qui fournit une marge largement suffisante pour accommoder toutes les bavures possibles. En outre, les techniciens qui fabriquent l'ordinateur suivent à la piste le cheminement des impulsions, et quand ils voient que celles-ci se sont trop dégradées, ils insèrent un organe de « remise en forme ». Cet organe est un circuit électronique qui engendre une belle impulsion vierge bien rectangulaire dès que quelque chose de discernable, fût-ce bossu ou crénelé, se présente à l'entrée, ce quelque chose ayant suffi pour affirmer qu'il y avait là une impulsion. Les circuits de régénération sont très faciles à faire et peu coûteux, et on en trouve répartis de-ci de-là plusieurs milliers dans chaque ordinateur. Grâce à quoi on peut dire que l'information *ne se perd ni ne se métamorphose jamais, elle ne devient jamais douteuse ou ambiguë*. Bref, l'interprétation du codage reste toujours certaine.

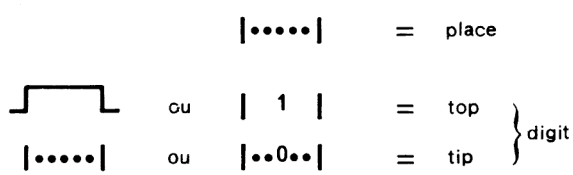
Au total, vous voyez qu'en adoptant le code binaire, la technique *ordino* accepte délibérément son caractère fruste et naïf pour bénéficier en revanche de la commodité d'usinage qu'elle permet, et surtout de la sécurité qu'elle confère à l'information au cours de ses voyages. Ce compromis se justifie pleinement si vous songez qu'un ordinateur moyen contient quelques centaines de milliers, ou millions, d'organes élémentaires, et que le traitement auquel il soumet l'information codée peut comprendre des centaines de milliers, ou millions, de pas successifs. Or si *un seul* pas est erroné, toute la suite devient fausse, et grossièrement fausse puisqu'une impulsion présente en une place où il n'en faudrait pas, ou absente d'une place où il en faudrait une, peut changer du tout au tout le sens de la configuration. Aussi la chasse aux erreurs, qui doit se faire ici avec une rigueur fantastique, constitue-t-elle la préoccupation majeure des techniciens, sur le même plan que l'astuce et la performance des circuits. Le code binaire assure qu'aucune erreur ne proviendra des impulsions mêmes qui incarnent les messages. D'autres sources d'erreur restent certes possibles, mais dont la cause ne réside pas dans la structure des signes codés. Elles proviennent de « coups de tête » des circuits, et nous verrons plus tard comment elles sont maîtrisées.

J'espère que vous avez maintenant adopté le codage binaire. Comme il est omniprésent dans un ordinateur, vous me permettrez d'introduire un peu de vocabulaire spécial pour n'avoir pas à traîner tout le temps des expressions aussi lourdes que « oui, impulsion électrique » ou « non, pas d'impulsion électrique ». Je vais vous proposer la terminologie suivante :

- s'il y a impulsion (« oui », ou « 1 », ou « trou »)
  on dira qu'il y a un ..... → **top**
- s'il n'y a pas impulsion (« non », ou « 0 », ou « plein »)
  on dira qu'il y a un ..... → **tip**

- pour désigner l'un *ou* l'autre de ces deux états,
  on emploiera le mot (emprunté à l'anglais) ..... → **digit** (*)
- pour désigner l'endroit aménagé pour y placer un **digit** (endroit dans l'espace s'il s'agit d'un carton qu'on perfore, ou dans le temps s'il s'agit d'une impulsion)
  on emploiera le mot ..... → **place**

Voici ces termes nouveaux rassemblés dans un tableau :



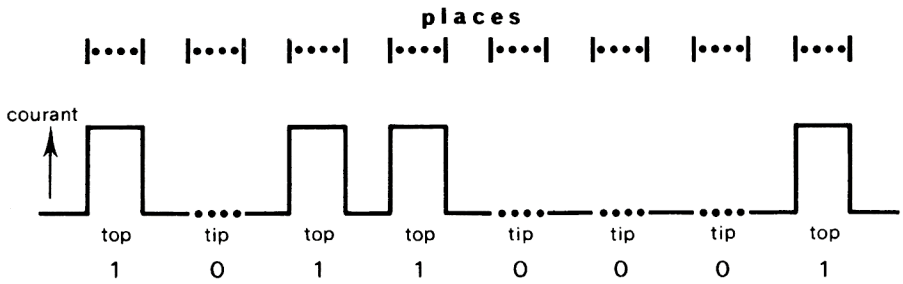
Vous voyez qu'on peut dire qu'un « top » est une *place pleine* ou « remplie du digit 1 », tandis qu'un « tip » est une *place vide* ou « remplie du digit 0 ».

Cette terminologie de tops et tips m'est personnelle. Beaucoup d'auteurs évitent de forger un vocabulaire spécial et se contentent de désigner les tops et les tips par les signes 1 et 0 employés dans leur représentation *homo*. Mais cela donne lieu à de fâcheuses embrouilles quand on emploie ces mêmes signes pour écrire des nombres.

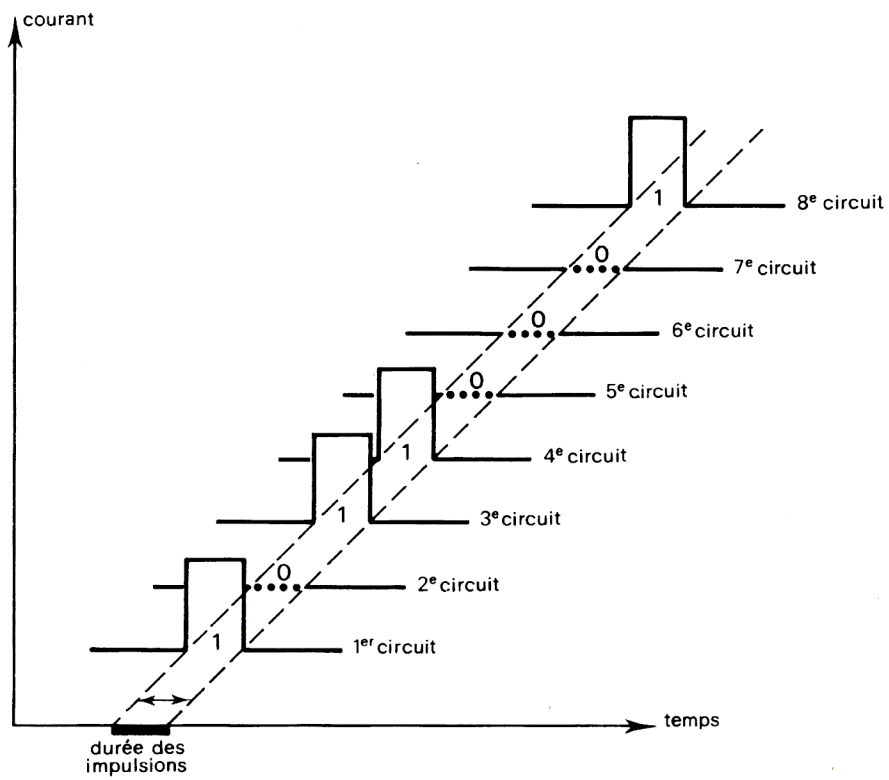
Vous n'oublierez pas, je vous prie, la grande règle selon laquelle un canevas de N *places* permet, en y distribuant des digits, c'est-à-dire des tops ou tips, de toutes les manières possibles, de former 2^N configurations différentes, donc de « dire », en binaire, 2^N choses (idées, concepts) différentes. Ainsi, comme $2^8 = 256$, on peut exprimer en code binaire 256 choses ou concepts avec 8 places.

N'oubliez pas non plus que, dans les ordinateurs modernes, l'information codée circule toujours en « parallèle ». Cela suscite une petite difficulté d'écriture. Pour représenter sur le papier un train de tops et de tips, tel que le train 10110010 de la figure **A** page 32, qui occupe 8 places, je ne peux faire mieux que de les dessiner côte à côte, de même que j'ai écrit, côte à côte, les 1 et les 0 de la représentation *homo* de ce train. Or, physiquement, un top est une impulsion de courant qui dure un certain *temps*, durée qu'illustre la largeur de sa base. De ce fait, on est entraîné à prendre toute la ligne de la base pour un axe des *temps*, et on interprète instinctivement la figure de type **A** comme une *succession* de tops et tips se déroulant *dans le temps*. Or, il n'en est rien. Les tops et les tips durent bien un temps égal à celui de leur base, mais dans la machine ils se produisent tous simultanément, sur 8 circuits parallèles — voyez la figure de type **B** où chaque circuit accommode l'une des 8 places, et les digits se présentent à la manière de chevaux courant de front sur 8 pistes parallèles.

(*) On emploie aussi le mot « bit », venu de la langue anglaise (« binary digit »). Mais il a été longtemps, et reste chez la plupart des auteurs, réservé à la numération binaire. J'ai donc préféré le mot « digit » qui est le terme générique, valable pour n'importe quel système de numération.



type A



type B

La figure de type **B** est donc la seule correcte, mais le dessin sur un papier se prête mal à une illustration de ce type. Il ne permet de représenter un train de digits que par une figure du type **A**, qui ne serait correcte que dans un ordinateur *série*. J'espère que vous me pardonnerez de m'abandonner, sur ce point, à la facilité. Vous êtes prévenu.

Codage des nombres

Dans l'exemple du mariage par lequel j'ai commencé ce livre pour stimuler votre imagination, nous codions des notions complètes telles que « très fortuné ». La voie logique est différente : elle consiste à établir d'abord un codage pour les *nombres et les lettres*, et à utiliser ensuite ce codage pour exprimer des idées plus élaborées par combinaison.

Le codage binaire des nombres ayant été expliqué dans un autre volume de cette collection (*), je me bornerai à vous en exposer rapidement l'ossature. Nous n'avons à notre disposition que *deux* chiffres, 1 et 0 (qui, en *ordino*, seront le top et le tip) et, en les faisant se succéder dans un ordre convenable le long d'une filière de places, nous devons représenter tous les nombres.

Suivant la tradition établie par le système de numération décimale que vous pratiquez tous les jours, l'écriture *homo* du système binaire procède de *droite à gauche*. La place la plus à droite d'une filière reçoit le rang 0. Puis, en allant chaque fois d'un cran vers la gauche, on trouve les places de rangs successifs 1, 2, 3, 4, 5, etc.

Dans l'unique place de rang 0 on ne peut écrire que les deux nombres 0 et 1, à savoir, bien entendu, le nombre 0 en remplissant cette place d'un tip et 1 en la remplissant d'un top. J'admets maintenant à côté à gauche la place de rang 1, où je place un top (si j'y avais placé un tip, c'est comme si je n'avais rien fait). La place de rang 0 (suivez la figure page 34 de haut en bas) pourra alors être occupée soit par un tip, ce qui donne la configuration 10, pour représenter le nombre 2, soit par un top ce qui donne la configuration 11 pour représenter le nombre 3. Puis je mets en service la place de rang 2, occupée par un top. De nouveau, j'utilise d'abord la place à droite que j'occupe d'abord avec un tip — d'où la configuration 100 du nombre 4, ensuite avec un top d'où la configuration 101 du nombre 5. Pendant ce temps la place intermédiaire de rang 1 n'avait pas servi. Je l'emploie maintenant en y plaçant un top, une fois avec tip à droite, donnant la configuration 110 du nombre 6, une seconde fois avec top à droite, donnant la configuration 111 du nombre 7. J'ai ainsi formé toutes les configurations possibles à trois places. J'en introduis une nouvelle, donc de rang 3, et je recommence le peuplement des places à droite comme tout à l'heure, par une règle que vous percevez aisément : dans le rang 0 l'alternance des tops et tips se fait à chaque passage d'un nombre au suivant, alors qu'elle ne se fait que de deux en deux dans le rang 1, de quatre en quatre dans le rang 2, et ainsi de suite. Vous vérifiez la grande règle qu'avec un total de N places disponibles, on peut représenter un maximum de 2^N choses différentes. Ainsi, avec un total de 4 places, on représente les $2^4 = 16$ nombres compris entre 0 inclus et 15 inclus. Il y a un petit déhanchement à cause du nombre 0, qu'il faut bien pouvoir représenter aussi. Ce déhanchement fait que le nombre 16 lui-même exige une cinquième place et se représente par la configuration 10000.

Ce fait que le nombre 16, qui est exactement la puissance quatrième de 2, est représenté en binaire par un 1 suivi de quatre zéros, n'est qu'un cas particulier d'un fait plus général, qui est d'ailleurs le seul que vous ayez à retenir, car vous n'êtes pas

(*) Voir, dans cette collection, le volume « Comprendre les mathématiques », chapitre 1.

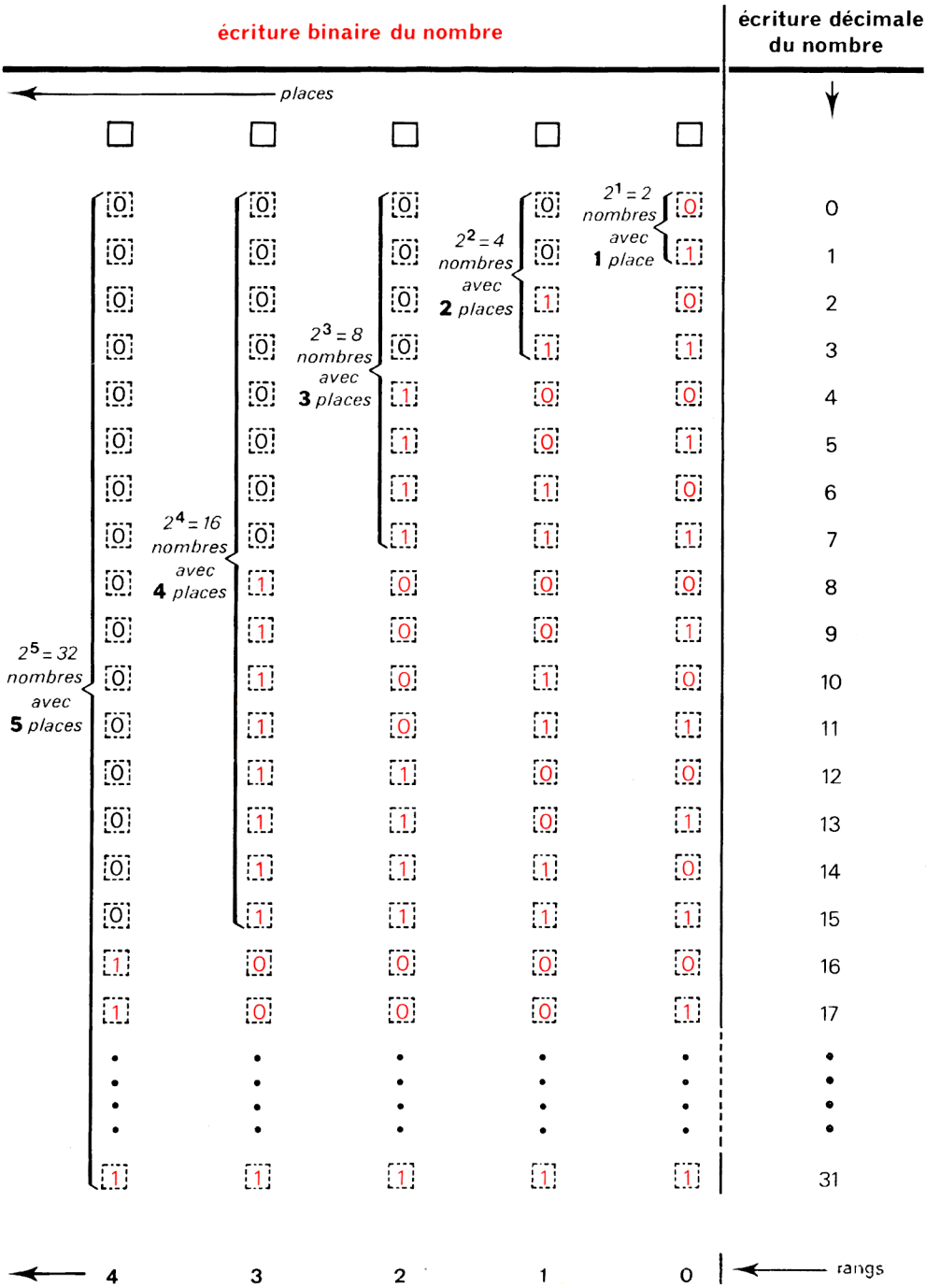
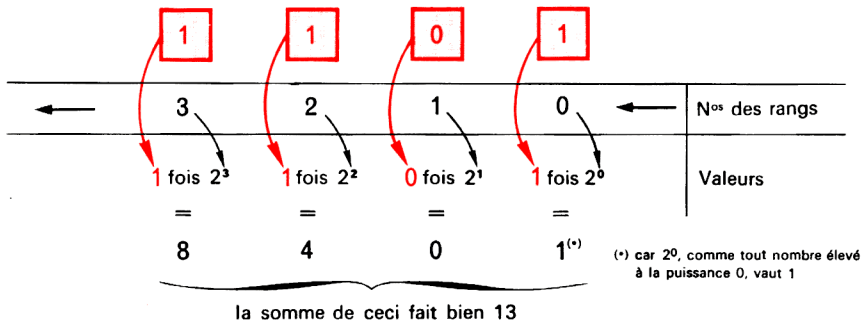
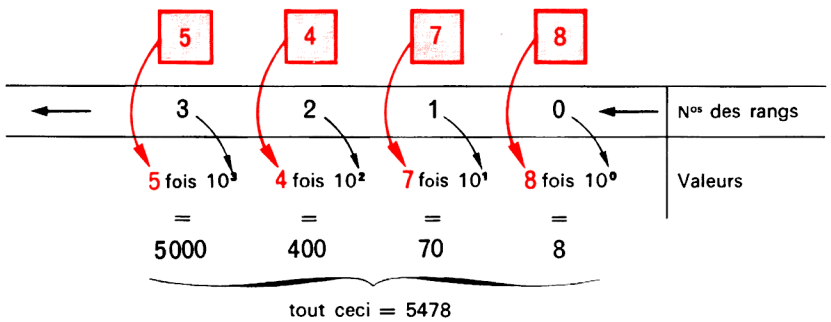


Tableau des nombres en binaire

destiné à écrire vous-même des nombres en notation binaire. Votre affaire est uniquement de *comprendre* cette notation et, à cet effet, ce que je vais vous dire maintenant contient tout ce qu'il est bon que vous sachiez. A savoir que chaque digit (top ou tip) placé au *rang* $\boxed{p}$ compte pour ce nombre de fois (donc soit 0 fois, soit 1 fois) la *puissance* $\boxed{p}$ de 2. Considérez par exemple la configuration binaire qui représente le nombre 13, d'après le grand tableau précédent :



D'ailleurs, la numération décimale qui vous est familière s'analyse exactement de la même manière, à ceci près que les diverses places ont pour poids les puissances de *10* égales à leurs *rangs*, au lieu des puissances de *2*. Voici un exemple en décimal :



C'est en vertu de ce principe général et fondamental de tout système de numération que, dans un système de base B, les puissances B^2, B^3, B^4, ... de la base s'écrivent avec $\boxed{1}$ en tête suivi de 2, 3, 4, ... zéros :

	signifie	
	en base 2	en base 10
10	2^1	10^1
100	2^2	10^2
1 000	2^3	10^3
10 000	2^4	10^4 etc.

En fait, pour rendre plus commodes la programmation et la circuiterie, on emploie rarement le système binaire pur tel que je viens de vous l'exposer. Le système de codage qui se répand de plus en plus en *ordino*, s'appelle « *décimal-binaire* » *et consiste à représenter en binaire, séparément chacun des chiffres de l'écriture décimale du nombre.* Comme la représentation binaire des deux plus forts chiffres décimaux, 8 et 9, est respectivement 1000 et 1001, c'est-à-dire exige 4 places, il faudra disposer d'autant de tranches de 4 places binaires qu'il y a de places décimales dans l'écriture décimale du nombre. Par exemple le nombre à trois places décimales 279 sera représenté comme suit dans trois tranches de 4 places binaires :

2	7	9
0 0 1 0	0 1 1 1	1 0 0 1

On peut se permettre de *joindre* les tranches, et d'écrire tout uniment 001001111001, si on s'est entendu avec le lecteur pour que ce soit *lui* qui prenne le soin de découper la filière en tranches de 4 places, et pour qu'il interprète le contenu binaire de chaque tranche comme signifiant *un* chiffre décimal.

Ce code a l'avantage que les personnes ou les machines ayant à l'appliquer n'ont besoin de connaître que les représentations binaires de dix nombres, à savoir les dix chiffres décimaux. Par contre, il conduit à des configurations plus longues, donc à un plus grand nombre de circuits. Mais le monde *ordino* moderne accepte facilement de multiplier les circuits si ça rapporte plus de sûreté ou de commodité. Cette humeur tient à des considérations économiques. Vous savez que les circuits des ordinateurs sont peuplés de ces petits cristaux miracles qu'on appelle des « transistors ». Or, quand ils apparurent sur le marché, vers 1954, les transistors valaient 15 francs pièce (15 francs de l'époque qui représentent bien 40 F de maintenant). Aujourd'hui, ils valent 40 centimes pièce, et encore les trouve-t-on de moins en moins isolés et de plus en plus intégrés par plusieurs dizaines à la fois dans des blocs monolithiques. Le prix a donc baissé de 100 fois pendant que la qualité s'améliorait de 100 fois. Cette évolution est la plus spectaculaire qu'ait jamais connue la technique humaine toutes branches comprises. Dès lors qu'on peut dire : « la transistaille, elle est pas chère » (comme les financiers disent « l'argent, elle est pas chère »), ce n'est pas la peine d'économiser sur le nombre des circuits si leur abondance simplifie ce qui devient plus cher, parce qu'impliquant davantage les recours à l'homme : la sécurité et la programmation. Le passage des anciens calculateurs série aux ordinateurs modernes parallèles s'inscrit dans la même évolution.

Mots et caractères

Le codage des nombres ne s'arrête pas tout à fait là. Je ne vous ai parlé que de nombres positifs plus grands que 1, du genre 2 705. Mais on peut aussi avoir des nombres plus petits que 1, du genre 0,000 027 05, et des nombres négatifs aussi bien que positifs. Pour accommoder toute cette gamme, on prend deux dispositions supplémentaires importantes.

D'abord, on consacre une place spéciale, généralement mise en tête, pour indiquer si le nombre qui va suivre est positif (on place alors dans cette place un top) ou négatif (on y place un tip). Ensuite, on convient d'écrire tout nombre sous la forme d'un produit, le produit de sa valeur *transportée dans l'intervalle* (0,1) par une puissance convenable de 10. Je m'explique par des exemples :

le nombre 2 705 000 000 000		peut s'écrire $(0{,}270\,5) \cdot 10^{13}$
le nombre	2 705	peut s'écrire $(0{,}270\,5) \cdot 10^{4}$
le nombre	2,705	peut s'écrire $(0{,}270\,5) \cdot 10^{1}$
le nombre	0,270 5	peut s'écrire $(0{,}270\,5) \cdot 10^{0}$ $(10^{0} = 1)$
le nombre	0,000 000 000 270 5	peut s'écrire $(0{,}270\,5) \cdot 10^{-9}$

(je vous rappelle qu'une puissance négative telle que 10^{-5} veut dire $\frac{1}{10^{5}}$, autrement dit $\frac{1}{100\,000} = 0{,}000\,01$). Vous voyez que ces six nombres, très différents par leurs valeurs mais contenant le même moule de chiffres significatifs, sont tous ramenés à un même nombre, dont la valeur est chaque fois comprise entre 0 et 1, multiplié par une puissance convenable de 10. L'exposant de cette puissance est simplement égal au nombre des chiffres lorsqu'il s'agit d'un nombre plus grand que 1; et, lorsqu'il s'agit d'un nombre plus petit que 1, l'exposant (ou plutôt sa valeur absolue puisqu'il devient négatif) est égal au nombre des zéros qui suivent la virgule avant le premier chiffre significatif. Cet exposant situe donc immédiatement la « magnitude » du nombre, comme disent les astronomes parlant des étoiles, ou encore, comme disent les physiciens, son « ordre de grandeur » : dizaines, milliers, centièmes, etc. Tandis que le moule donne l'anatomie fine du nombre.

On prend dès lors le parti d'indiquer *séparément*, d'une part, les chiffres composant le moule — chacun écrit en binaire, bien sûr, lorsqu'on fait de l'*ordino* — et d'autre part, la valeur de l'exposant, également écrite en binaire et précédée de son propre signe.

C'est dans le moule que niche la « valeur précise », et à ce propos il importe que je vous dise un mot. Ni la feuille de papier du monde *homo*, ni la mémoire du monde *ordino*, ne disposent d'une infinité de places. On ne peut donc pas écrire des moules aussi fournis qu'on veut, comme par exemple : 0,270 539 610 082 540 328 910 044... Aussi n'en a-t-on nul besoin, car aucune grandeur n'est jamais définie ni définissable avec tant de précision. Après un certain rang, les chiffres qu'on alignerait n'auraient en général aucune signification, et on peut, on doit s'en passer. Il est bien rare qu'on puisse donner un sens à plus de six chiffres décimaux, dans un moule. Dire, par exemple, qu'une longueur vaut 834 071 mètres, soit $(0{,}834\,071) \cdot 10^{6}$, avec un moule comprenant 6 chiffres, cela impliquerait qu'on est sûr que le sixième est bien 1, et non 2, c'est-à-dire qu'on est sûr de la longueur à un millième de millimètre près! C'est la limite des possibilités humaines en matière de précision.

De même qu'on n'atteint jamais des précisions infinies, ce qui limite l'étendue des places à prévoir dans les moules, de même on ne rencontre jamais des ordres de grandeur infinis, ni dans le sens de l'immensité ni dans celui de la petitesse, et cela permet de limiter la valeur maximum qu'on offre à l'exposant. La plus grande longueur connue, celle qui nous sépare de la plus lointaine nébuleuse, fait mille milliards

d'années-lumière, et s'exprime, en mètres, par 10^{28}. C'est que les puissances, ça croît fantastiquement vite! Et les puissances négatives décroissent de même. La plus petite longueur connue, qui est celle de certaines ondes de rayons dits « γ », s'exprime, en mètres, par 10^{-20}. Vous voyez qu'en prévoyant que les exposants pourront aller jusqu'à $\pm$ 64, on est largement paré pour toutes les éventualités en ce qui concerne la magnitude des nombres.

Il résulte de cette discussion qu'on peut accommoder à peu près tous les nombres qu'on aura des chances de rencontrer avec une filière de 32 places organisée comme suit :

— la première place à gauche sert à indiquer le *signe du nombre* (par exemple on y mettra un top pour signifier que le nombre est positif, un tip pour signifier qu'il est négatif);

— la seconde place à partir de la gauche sert à indiquer le *signe de l'exposant* (même convention, top pour exposant positif, tip pour exposant négatif);

— les 6 places suivantes sont consacrées à la *valeur absolue de l'exposant*, en binaire pur, ce qui permet d'aller de 0 inclus à 63 inclus;

— les 24 places suivantes servent à accommoder *le moule*, dans 6 tranches de 4 places, indiquant chacune, en binaire, un chiffre décimal, donc au total 6 chiffres décimaux pour le moule.

Beaucoup de machines allouent en effet une filière de 32 places à l'accommodation des nombres. Une telle filière s'appelle en informatique un **mot** de l'ordinateur et, par un abus coupable mais irrésistible, on désigne ainsi à la fois le *cadre* de 32 places préparé pour être rempli ***et*** *l'information* qu'on inscrit dedans. C'est comme si on appelait « train » à la fois les wagons et l'ensemble des voyageurs qui s'y sont installés. Les informaticiens sont assez intelligents, et supposent que vous l'êtes aussi, pour discerner d'après le contexte dans lequel de ces deux sens doit s'entendre leur « **mot** ».

Jadis, le « **mot** » était la brique de base *uniforme* d'un ordinateur. Mais quand on sut utiliser un même ordinateur pour des problèmes très variés, on s'aperçut que, dans le traitement de certains problèmes, les mots étaient à peu près tous remplis, tandis que dans le traitement d'autres classes de problèmes, les mots restaient à moitié vides. Le second cas implique un certain gâchis. Comme on acquérait en même temps une puissance toujours plus grande pour réaliser des manœuvres toujours plus astucieuses, on prend aujourd'hui de plus en plus le parti d'accepter pour les « mots » une longueur un peu *flottante*. L'unité de taille vraiment « insécable » de l'information *ordino* tend à s'appeler (mais le vocabulaire sur ce point n'est pas encore fixé) le **caractère,** et ne contient, par exemple chez IBM, que 8 places — ce qui fait qu'on a baptisé « **octet** » le support d'un tel caractère. On peut alors composer des **mots** avec un nombre plus ou moins grand de **caractères,** selon le besoin. Mais ce sont là des raffinements de spécialistes dans lesquels vous n'avez pas à entrer. Pour l'intelligence que vous voulez acquérir de l'informatique, il suffira, et il vous sera plus agréable, d'imaginer que dans l'ordinateur ne circulent que des ***mots*** *de longueur fixe*, longueur qu'on pourra appeler leur **taille,** et qui pourra accommoder plusieurs **caractères.** Nous considérerons le plus souvent des mots de 32 places pouvant comprendre 4 caractères de 8 places ou octets.

Codage des lettres et signes

Ce paragraphe sera extrêmement court, car l'idée qu'il expose est on ne peut plus simple. Elle consiste en ce que les lettres de l'alphabet et divers signes typographiques courants, tels que les signes de ponctuation, les symboles désignant le dollar, le mot « compagnie », ou des opérations arithmétiques, sont tout aussi codables en binaire que les nombres. C'est même par ce codage-là que nous avons commencé le présent livre. On s'est aujourd'hui entendu sur la table complète des signes de ce genre qu'un ordinateur doit savoir traiter : il y en a 64, et les graphismes par lesquels on les représente en *homo* sont aussi à peu près standardisés. Quant à la correspondance entre ces signes et les 64 configurations de tops et tips qui les représentent, elle diffère hélas de constructeur à constructeur. Pour représenter les 64 signes il faut, comme vous savez, *6* places car 64 est égal à 2^6.

C'est pourquoi, pendant longtemps, le « caractère » de base des ordinateurs fut une filière de *6* places. Mais ça ne permettait pas cet emploi polyvalent de la filière que préfèrent les techniciens modernes, en particulier, pour accommoder des nombres. En effet, 6 places sont de trop pour la représentation binaire d'*un* chiffre décimal — qui n'exige que 4 places — et trop peu pour *deux* chiffres décimaux. D'où l'emploi de l'*octet* comme filière de base.

Il n'y a rien de plus à dire sur le codage des signes, sujet évident. Par contre, le codage des **instructions,** sujet dans lequel nous entrons maintenant, est plus difficile. Je m'en excuse, mais il importe que vous le compreniez, car c'est l'un des piliers secrets du palais *ordino*.

Les instructions et leur codage — Première idée

Lorsque vous avez lu qu'un ordinateur n'est qu'une machine pour exécuter très vite la suite d'actes élémentaires que lui prescrit un programme, vous vous êtes peut-être demandé — vous auriez dû vous demander — mais où donc siège, physiquement, le programme? Si l'homme qui l'a conçu ne pouvait faire mieux que de l'inscrire sur des feuilles de papier, sous la forme *homo* suivante :

1er pas : faire ***telle*** chose avec ***telle*** donnée,

2e pas : faire ***telle*** chose avec ***telle*** donnée,

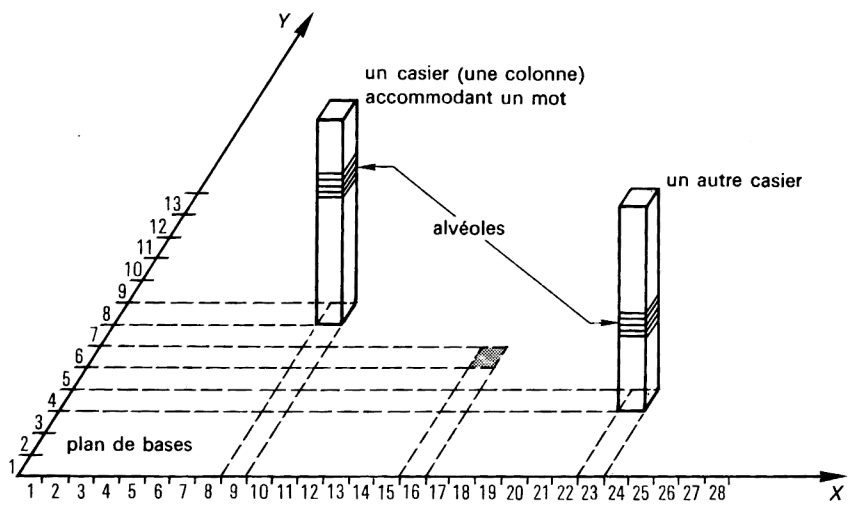
3e pas : faire ***telle*** chose avec ***tel*** résultat,

4e pas : etc., jusqu'au 1 000e ou 10 000e pas,

alors on ne serait pas sorti de l'auberge. Car il faudrait transmettre chaque ordre à la machine à chaque pas, en le transposant du langage *homo* en langage *ordino*. Et avec la plupart des ordres, il faudrait dire encore à la machine sur quelle donnée elle doit accomplir le travail qu'on lui ordonne. Et comment ferait-on lorsque les données à traiter à un certain pas sont, non pas celles du départ que je connais, mais les résultats intermédiaires d'opérations exécutées dans la machine, et que je ne connais pas? Sans compter que la plupart des programmes contiennent de nombreuses répétitions de grands tronçons de programmes comportant des suites de pas identiques. Va-t-on devoir répéter les instructions à la machine à chaque boucle?

Il y a là un problème de base, sans la solution duquel la belle notion d'un esclave mécanique exécutant mon traitement reste creuse et irréelle.

Cette solution, vous l'avez sans doute deviné, consiste à *loger le programme lui-même dans la mémoire de l'ordinateur*. Mais qu'est-ce que cela veut dire pratiquement? Une mémoire *ordino*, nous le verrons plus tard en détail, est un réseau de « ***casiers*** ». *Chaque* ***casier*** accommode un « mot », et se décompose donc en autant de *cellules, ou alvéoles*, que le mot contient de places, par exemple, 32. Spatialement, représentez-vous un casier sous la forme d'une petite *colonne* dressée verticalement sur une base carrée et faite de 32 alvéoles cubiques empilées exactement l'une sur l'autre. Dans une mémoire d'un ordinateur il y a quelques dizaines de milliers de tels casiers — colonnes tassées les unes contre les autres. Les bases de ces colonnes forment donc un carroyage régulier sur un « plan de base ». Pour repérer une colonne particulière il suffira de repérer sa base dans ce plan, et pour cela on emploiera naturellement le système bien connu des deux coordonnées cartésiennes sur deux axes OX, OY qu'on aura subdivisés en segments ayant chacun la longueur d'un côté de la base, et numérotés 1, 2, 3, ... sur OX et de même 1, 2, 3, ... sur OY. Dès lors chaque casier est repéré par le couple des coordonnées X, Y, de sa base, couple qu'on appelle son « **adresse** ». Par exemple, les deux casiers-colonnes de la figure ont pour adresses respectivement (9, 8) et (23, 4). A l'intérieur d'un même casier, les diverses alvéoles sont repérées naturellement par leurs rangs, mais ce repérage concerne le **contenu** d'un casier. Quand on se borne à désigner un **contenant,** il suffit de donner son **adresse** par un couple (X, Y) sans mentionner telle ou telle alvéole, l'adresse étant valable pour toute la colonne, c'est-à-dire tout un mot.



Lorsqu'ils sont entre eux, les informaticiens, avec la négligence du vocabulaire qui les caractérise, disent « **adresse** » à la fois pour désigner l'*endroit* où s'élève un casier de la mémoire, *et* le casier lui-même. Les moins soigneux vont jusqu'à appeler également « adresse », le *mot contenu dans le casier*, comme si on disait, parlant d'un quidam, « Monsieur 197 rue de Vaugirard ». Peut-être m'arrivera-t-il aussi de céder à cette licence de langage. Que Dieu me pardonne!

Les instructions et leur codage — Deuxième idée

Nous supposons maintenant que la machine est organisée de telle manière que si je lui fournis une adresse, c'est-à-dire deux nombres X et Y, elle achemine vers cette adresse des impulsions de courant qui, par un arrangement que nous verrons plus loin, permettent soit d'*inscrire* un mot dans le casier situé à cette adresse, soit de *lire* le mot qui y était inscrit depuis la dernière fois. Je prétends que ceci nous permet de « mettre le programme en mémoire » et de le faire ensuite dérouler et exécuter automatiquement par la machine. Il y faut une idée de plus.

▶ — Voyez-vous laquelle ?

▶▶ — J'avoue que non.

▶ — Réfléchissons. Dans la mémoire *ordino* ne siègent que des mots binaires faits de tops et de tips. Or, ce que je prétends quand je dis que je vais y inscrire le *programme*, c'est que la mémoire contiendra des *instructions*. Que faut-il que je fasse pour connecter ces deux notions ?

▶▶ — Je vois. Vous voulez sans doute me suggérer de *coder aussi les instructions* sous forme de mots binaires, en tops et tips ?

▶ — Exactement. Rappelez-vous qu'une instruction — on dit aussi un « ordre » — est l'énoncé de l'un des actes élémentaires qu'on prescrit à la machine d'exécuter. Coder ces instructions veut donc dire qu'à chacun de ces actes nous attachons un *nombre*, en binaire, qui le représente. Par exemple, l'instruction (ou l'ordre) « transfère », qui devra amener la machine à transporter une information d'un endroit à un autre, se notera, mettons, 0000001 ; l'ordre « recopie » se notera 00000010 ; l'ordre « efface » se notera 000000011 ; l'ordre « additionne » se notera 00000100 ; l'ordre « compare » se notera 00000110 ; etc. L'ensemble de ces nombres constitue ce qu'on appelle le « **code d'ordres** » de la machine. Vous ne les confondrez pas s'il vous plaît avec les nombres proprement dits, qui n'ont pas besoin d'être décodés pour qu'on sache ce qu'ils signifient, car ils jouent leur rôle de nombre. D'ailleurs, les « nombres-nombres » peuvent occuper toute la gamme des milliards de milliards de milliards ... de valeurs qu'accommode l'ordinateur (par exemple depuis 10^{-64} jusqu'à 10^{64}), tandis que des « nombres-ordres » il n'y en a qu'autant qu'on a besoin d'ordres distincts pour faire exécuter toutes ses besognes à la machine. Et nous verrons que cet éventail est infiniment plus restreint, pour la raison évidente que les besognes les plus complexes peuvent toujours être décomposées en une suite de quelques actes élémentaires fondamentaux peu nombreux.

Quel est le plus petit nombre d'actes élémentaires différents avec lesquels on peut assurer l'exécution des tâches les plus lourdes ? La question n'admet pas de réponse définie. A un idiot de village, on peut faire faire des tas de choses avec seulement une demi-douzaines de gestes, pour peu qu'on s'arme de patience. De même, toutes les couleurs de la nature pourraient être reconstituées en mélangeant seulement les trois couleurs de base de Maxwell, mais il faudrait pour cela mesurer leurs doses avec trois décimales. Pour se faciliter la tâche, les peintres prennent donc des palettes plus fournies, comprenant de 10 à 20 couleurs de base, au lieu de 3. Un ordinateur très bête, qui ne saurait qu'additionner, pourrait aussi multiplier, par exemple X par 3 258, en ajoutant 3 258 fois X à lui-même.

En règle générale, plus on enrichit la palette des ordres différents qu'un ordinateur sait exécuter, plus on augmente la rapidité de ses traitements; et chaque constructeur se vante d'avoir mis au point la palette d'ordres la plus astucieuse, c'est-à-dire donnant à la machine un maximum de puissance avec un minimum d'ordres différents. La plupart des ordinateurs disposent d'une gamme de moins de 100 ordres différents, avec lesquels ils se débrouillent fort bien. L'expérience prouve que les plus grosses palettes d'ordres auxquelles on puisse songer n'en auront sûrement pas plus de 256. Or, 256 est égal à 2^8, et cela veut dire que, pour accommoder les 256 nombres représentant ces ordres, il suffit de ..... de combien de places binaires?

▶▶ — De 8, ou un octet.

▶ — Donc d'une filière nettement plus petite que la *taille* des mots, ces mots conçus pour contenir des nombres vrais, je veux dire qui ne servent pas à représenter des ordres mais jouent bien un rôle de nombres. Nous avons vu en effet que cette taille comprend 32 places, ou 4 octets! Dès lors, vous voyez poindre une subdivision des mots en deux grands genres. Il y aura les *mots dits* « **numériques** » qui accommodent des « *nombres-nombres* » dans leurs 4 octets, comme je vous l'ai déjà expliqué. Et d'autres mots que nous appellerons « **logiques** » et dans lesquels un octet (en général le premier à gauche) accommode au contraire les « *nombres-ordres* ».

▶▶ — Et qu'est-ce qu'on met dans les trois autres octets d'un « *mot logique* »?

▶ — Que proposeriez-vous qu'on y mît?

▶▶ — On pourrait peut-être y inscrire la donnée à laquelle doit s'appliquer l'ordre exprimé dans le premier octet?

▶ — L'idée paraît tentante en effet, mais, elle se heurte à deux difficultés. La première, mineure, est qu'une fois qu'une tranche d'un mot est occupée par un nombre-ordre, la place qui reste ne permet plus d'y inscrire une donnée, laquelle s'exprime par un nombre-nombre exigeant la taille entière du mot. La seconde difficulté de votre proposition est majeure, et même dirimante : c'est que la plupart des données à traiter aux divers pas d'un programme *ne sont pas connues a priori du programmeur*! Car elles émanent du traitement lui-même! Ici se place une réflexion si fondamentale de l'entreprise *ordino* que je n'hésite pas à vous la développer en détail, en m'appuyant sur une analogie du monde *homo*.

La valse des adresses

Beaucoup de vues techniques bouleversantes ont pour origine une réflexion neuve sur des activités humaines devenues si routinières qu'on avait cessé de percevoir leur trame intelligente et fine.

Regardez de plus près comment s'exécute, sous votre propre main, une chaîne d'opérations arithmétiques très simples, telle que

$$\frac{(a \times b) + c}{d}$$

où a, b, c, d sont des nombres donnés.

Vous faites d'abord ($a \times b$), sans savoir d'avance le résultat, car autrement vous n'auriez pas besoin de travailler. Ce résultat, que je vais appeler p, vous le posez, une fois obtenu, en un endroit de votre papier, endroit que je vais appeler N° 1. Quand vous devez ajouter ce résultat $p = (a \times b)$ à c, votre regard va le chercher dans cet endroit N° 1, puis vous exécutez l'opération $p + c$. Cette opération donne un autre résultat, que je vais appeler q, que vous ne connaissiez pas non plus d'avance. Vous posez q dans un second endroit du papier que je vais appeler N° 2. Quand, au dernier pas, vous avez à diviser ce $q = (a \times b) + c$ par d, votre regard va chercher la valeur de q dans cet endroit N° 2.

Vous constatez que la lecture des divers endroits du papier où vous avez inscrit des données transitoires, bien qu'elle s'effectue à l'œil si vite que vous n'y prêtez pas attention, joue en réalité un rôle important dans l'affaire. Ce rôle devient crucial si l'opération ne doit pas être exécutée par vous-même, mais par quelque secrétaire agile qui ne sait *a priori* rien de vos intentions et doit agir en votre absence. Vous devez alors lui laisser des instructions. A cet effet, le seul moyen est de systématiser l'usage des endroits du papier pour inscrire ce que la secrétaire agile devra faire, et de lui dire dans votre message à quels endroits elle devra se reporter, comme le faisait à l'instant votre œil.

Pour mieux faire le pont entre cette situation *homo* et la procédure *ordino*, remplaçons le papier et ses endroits numérotés par un bloc de tiroirs, ayant chacun une adresse, et dans lesquels nous déposons des billets. La secrétaire agile dispose d'une petite « usine » (mettons une simple machine à calculer de bureau mécanique) pour faire exécuter les opérations qu'on lui commande. Appelons enfin « **vade-mecum** » le message qui va la diriger, conçu par le programmeur.

L'idée-force que nous avons dégagée en examinant l'exécution *homo* d'un programme, c'est que ce vade-mecum ne contient pas du tout, comme on pouvait le croire, la description même des opérations à faire! Il contient uniquement *une suite d'adresses de tiroirs*! Que fait avec ça la secrétaire agile?

Elle lit la *première* adresse écrite sur son vade-mecum, va au tiroir correspondant, l'ouvre, et y trouve un billet sur lequel le programmeur avait inscrit quelque chose à faire. Si ce quelque chose consiste à opérer sur deux nombres, le billet contiendra-t-il, d'après vous, les valeurs des deux nombres?

▶▶ — Non, pas du moins si on est en *ordino*, car je devine que dans ce cas, votre tiroir devient un casier de la mémoire, et vous m'avez dit que dans un casier ne siège qu'un mot, et que dans un mot il n'y a pas la place pour accommoder à la fois un ordre et les données auxquelles il faut appliquer l'ordre. J'en déduis que si le billet que sort la secrétaire agile l'instruit de faire quelque opération sur des nombres, les valeurs de ceux-ci ne peuvent pas figurer sur le billet même.

▶ — C'est parfaitement raisonné. Mais alors, comment la secrétaire agile apprendra-t-elle ces valeurs?

▶▶ — C'est facile. Le billet, outre l'ordre qu'il lui donne, l'envoie à l'*adresse d'un autre tiroir*, où elle trouvera un second billet qui porte cette valeur.

▶ — Et ce second billet dispose, lui, d'une place suffisante, puisqu'il ne contient plus qu'une valeur de nombre, d'un nombre-nombre et non d'un nombre-ordre. Le billet que la secrétaire agile trouve dans le *second* tiroir, porte ce que nous avons

appelé un « *mot numérique* », tandis que celui du *premier* tiroir, portait ce que nous avons appelé un ..... un quoi?

▶▶ — Un « *mot logique* ».

Vous voyez maintenant de façon claire en quoi ces deux genres de mots diffèrent, bien qu'ils soient tous deux formés de tops et de tips. Dans un mot *logique*, le premier octet de tops et tips contient la représentation codée d'un *ordre* et les autres octets contiennent les coordonnées d'une *adresse*. Au contraire, dans un mot *numérique* les tops et tips de tous les octets forment simplement un *nombre unique*, c'est-à-dire la valeur d'une des données sur lesquelles on opère. Les trois octets d'un mot logique qui ne sont pas occupés par un nombre-ordre suffisent largement à accommoder une adresse (X, Y) car ils comprennent 24 places, soit 12 pour X et 12 pour Y. Or, avec 12 places on peut désigner $2^{12} = 4\,096$ valeurs, et une mémoire ayant 4 096 valeurs de X et 4 096 valeurs de Y contient $(4\,096)^2$, soit environ 16 800 000 casiers. On ne connaît pas encore de mémoire si grosse, de sorte que 32 places suffisent largement pour un mot logique fait d'un ordre et d'une adresse.

L'ossature fondamentale d'un ordinateur

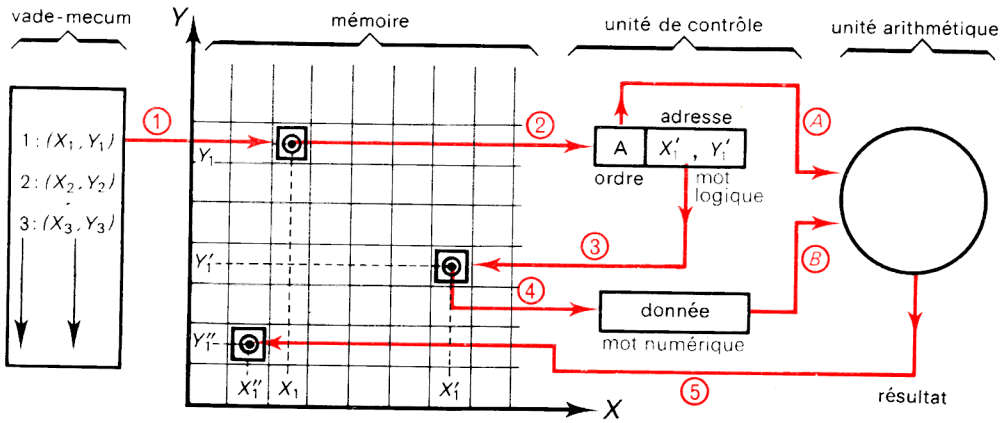
Retournant à notre secrétaire agile, vous voyez que son travail consiste à *voltiger d'adresse de tiroir en adresse de tiroir*. Dès qu'elle a rassemblé tous les matériaux nécessaires à une opération, elle les fait entrer dans sa petite usine en actionnant tous les boutons nécessaires pour que celle-ci exécute le traitement prescrit. Lorsque la petite usine a terminé sa besogne, elle délivre un autre billet portant le résultat qu'elle a obtenu. Ce billet, qu'est-ce qu'on en fait?

▶▶ — On le place dans un nouveau tiroir.

▶ — Et notez — détail capital — que le programmeur *n'a pas besoin de connaître ce résultat*! L'important est qu'il ait prévu de le faire déposer dans un tiroir bien déterminé, de façon que la secrétaire agile sache où elle pourra le trouver lorsque le moment sera venu de lui faire jouer un rôle.

La figure ci-après illustre ce mécanisme. Comme il est assez alambiqué, je vous conseille de le récapituler avec moi en suivant les cheminements du doigt. J'ai transposé l'analogie *homo* qui m'a servi jusqu'ici dans le monde *ordino*, comme vous vous y attendiez. Vous aviez correctement deviné que l'équivalent en *ordino* du bloc des tiroirs est la mémoire. Ce que j'ai appelé « la petite usine » s'appelle, en *ordino*, l' « *unité arithmétique* », en souvenir de l'époque héroïque où l'ordinateur n'effectuait que des opérations arithmétiques. Un nom plus correct par rapport à l'ordinateur moderne serait « *unité de traitement* », encore qu'on pourrait arguer que dans un ordinateur tout est traitement, y compris une inscription dans, ou une extraction de, la mémoire. Enfin l'équivalent de la secrétaire agile elle-même, qui lit et interprète les billets et prépare en conséquence la petite usine, s'appelle, en *ordino*, l' « *unité de contrôle* ». Ce sont les désignations que vous trouverez sur la figure. Le cheminement que nous allons y suivre est celui formé par les lignes en couleur.

La première indication du vade-mecum est une certaine adresse (X_1, Y_1) où il faut aller voir. On y va par la flèche ①. On trouve là un casier (marqué par son



pourtour carré) dont le contenu (dans le rond) est porté par la flèche ② sous le regard de la secrétaire — pardon, vers les circuits d'identification de « l'unité de contrôle ». Cette unité constate que ledit contenu est un mot logique comportant un ordre A et une seconde adresse (X'_1, Y'_1). L'unité conserve l'ordre sur une ligne *A*, pendant qu'elle expédie par la flèche ③ un signal d'exploration vers cette seconde adresse (X'_1, Y'_1). Il en sort par le fil ④ un mot qui est cette fois numérique, c'est-à-dire exprime une donnée. L'unité de contrôle, l'ayant constaté, accommode cette donnée sur la ligne Ⓑ qui fait entrer la donnée dans la petite usine — pardon dans l'unité arithmétique — au même moment où y entre aussi par la ligne Ⓐ l'ordre prescrivant ce qu'il faut faire avec cette donnée. Le résultat est inscrit, via la flèche ⑤, dans un casier de la mémoire désigné d'avance par le programme, casier d'adresse (X''_1, Y''_1). Ceci achève le cycle des manœuvres à exécuter au premier pas du programme. Le vade-mecum annonce alors le 2e pas, et un cycle analogue commence.

Il ne reste plus qu'une petite disposition à prendre pour tout automatiser : c'est de *loger le vade-mecum lui-même dans la mémoire*! Afin de ne pas multiplier indéfiniment le jeu des tiroirs qui se passent le mot les uns aux autres, on décide de consacrer au vade-mecum une zone à part de la mémoire, en plaçant les adresses successives (X_1, Y_1), (X_2, Y_2), (X_3, Y_3) dans des casiers *côte à côte* qui eux-mêmes se succèdent dans l'espace comme les numéros d'ordre 1, 2, 3, ... des pas du programme. Il suffira dès lors, pour tout déclencher, qu'il y ait quelque part une sorte d'adjudant faisant l'appel successif des numéros des pas : 1, 2, 3, ... etc. Quand il a dit « *1* », le *premier* casier de cette zone spéciale de la mémoire s'ouvre automatiquement, délivre l'adresse (X_1, Y_1) que le programmeur y avait inscrite, et le cycle précédent démarre. Quand ce cycle est achevé par le dépôt du résultat dans le casier (X_1, Y_1), le récitant appelle automatiquement « *2* » et immédiatement c'est le *deuxième* casier de la mémoire qui livre son contenu, c'est-à-dire une adresse (X_2, Y_2) qui déclenche un second cycle semblable. Et ainsi de suite. Nous verrons au chapitre du hardware comment cet adjudant récitateur est réalisé.

Ce n'est qu'avec ce dernier trait — l'insertion du vade-mecum dans les casiers à la queue-leu-leu d'une zone spéciale de la mémoire dont les numéros sont appelés successivement — que s'achève l'œuvre à l'édification de laquelle nous nous étions

attelés voici une dizaine de pages, et qui consistait à « loger le programme dans la mémoire » pour qu'il puisse se dérouler et être exécuté automatiquement. Peut-être ce projet vous a-t-il paru tout simple au moment où je vous le proposais? Vous constatez qu'il n'en était rien.

Un peu de genèse

En fait, la conception claire du mécanisme que je viens de vous exposer et qu'on peut appeler la « *marche par adresses* », était si délicate qu'elle ne fut mise au point définitivement, par le grand von Neumann, que plusieurs années après qu'on eût construit les premiers calculateurs électroniques. Pendant ces années de leur première enfance, les calculateurs faisaient déjà les chiens savants, au grand étonnement des profanes, en exécutant par voie électronique des additions et multiplications en quelques millièmes de seconde. Mais les initiés savaient que ce miracle, tout seul, ne servirait à rien tant qu'on ne saurait pas coudre ensemble une série longue et cohérente de ces opérations, de façon que la série se déroule toute seule sans que la machine ait à revenir à tout moment vers le demandeur du service. Le codage des instructions sous forme de « nombres » et la « marche par adresses », qui peuvent sembler aujourd'hui les solutions évidentes à ce problème, apparurent aux pionniers des temps héroïques comme des phares dans la nuit, et la littérature de l'époque appelle ces deux idées, avec tout le respect qu'on témoigne aux grandes inventions, le « plan von Neumann ».

Vous comprendrez mieux le rôle de ce « plan » dans la genèse des ordinateurs si vous réfléchissez à la sensationnelle souplesse qu'il confère à la machine. En codant les ordres sous forme de nombres binaires, on peut non seulement les faire entrer en mémoire à l'instar des données, mais encore *opérer* dessus comme s'ils étaient vraiment des nombres. Si par exemple l'addition est représentée par le « nombre » A et la multiplication par le « nombre » M, je peux passer de l'une à l'autre en *ajoutant* à A la différence $M - A$, calculée d'après les valeurs numériques de M et A. Quant à la « marche par adresses », elle permet non seulement de programmer d'avance de longues suites d'actions sans avoir à connaître les résultats intermédiaires, mais encore d'introduire dans les programmes une foule de tactiques extraordinairement puissantes. Vous apprendrez à en connaître plusieurs par la suite, mais je vais vous en citer une dès maintenant pour vous faire apprécier la portée du « plan von Neumann ».

Les travaux à faire sur ordinateurs comprennent très souvent la répétition d'un cycle d'opérations identiques où seules les données changent à chaque cycle. C'est le cas par exemple quand on fait l'état des comptes des déposants dans une banque : on prend chaque fois un autre compte, mais le programme pour en dresser le bilan reste exactement le même. Dans d'autres cas, on cherche des solutions à un problème par le procédé dit des approximations successives, qui conduit à recommencer un cycle de calculs toujours les mêmes en prenant chaque fois pour données de départ les valeurs auxquelles avait abouti le cycle précédent, jusqu'à ce qu'une certaine erreur résiduelle soit jugée suffisamment « petite ». C'est ainsi que pour extraire le sable d'un tas de terre, on repasse à travers un tamis de plus en plus fin le résidu d'un tamisage antérieur.

La « marche par adresses » permet alors d'écrire une « boucle de programme » *unique*, ne contenant que les adresses où il faut chercher la pâture de l'action, en prévoyant simplement un ordre spécial qui, lorsqu'un cycle est terminé, envoie ses données finales *aux mêmes adresses* où logeaient les données initiales de ce cycle, en chassant ces dernières (ôte-toi de là que je m'y mette); et le cycle, automatiquement, recommence à dérouler les mêmes pas sur les données nouvelles.

Je vous prie de noter soigneusement le trait important que voici : le « plan von Neumann » ne contient *aucune invention technique*. Les moyens matériels pour traiter l'information selon ce plan sont disponibles et simples. Ce ne sont pas eux qui posaient le problème. S'il a été résolu, c'est seulement par une réflexion, mais combien profonde, sur le « *modus operandi* ». Et cette réflexion elle-même ne *bouleversait pas* les vieux chemins de l'homme, au contraire, elle ne faisait que les creuser davantage. Bien des progrès n'ont ainsi résulté que d'une meilleure compréhension de ce qu'on fait tous les jours.

On a vu de plus près comment on peut coder les nombres et les lettres au moyen d'un canevas de *N places* disposées côte à côte et en chacune desquelles on fait apparaître ou ne pas apparaître une *impulsion* de courant électrique de forme standard. Chaque nombre, ou chaque lettre, est caractérisé par un remplissage particulier du canevas avec des impulsions ou des vides, qu'on appelle aussi des « oui » et des « non », ou encore des « tops » et des « tips », et mathématiquement des « 1 » et des « 0 ». La limitation à deux des états qu'on inscrit dans chaque place fait dire que le codage est *« binaire »* et vous verrez qu'en dépit ou à cause de son apparente candeur, cette méthode permet d'éviter les erreurs d'interprétation. Un canevas de *N* places remplies de tops et tips (de « 1 » ou de « 0 ») s'appelle un *mot binaire*. Quand un tel mot est déposé dans une mémoire, il y occupe une *case* formée de *N cellules*. De là on passe à l'idée féconde de coder aussi en mots binaires non seulement les informations, mais encore — d'une part, les *instructions* (ou *« ordres »*) que l'ordinateur est capable d'exécuter, ce qui constitue son *« code d'ordres »* — d'autre part, les *adresses* des cases de mémoire. Un mot qui contient les codes d'un ordre et d'une adresse de mémoire s'appelle un *« mot logique »* par opposition aux *« mots numériques »* qui contiennent les valeurs des données. Dès lors nous pouvons instruire l'ordinateur d'exécuter de longues suites d'opérations élémentaires sans que nous ayons besoin de connaître les résultats intermédiaires. Vous avez appris que von Neumann imagina à cet effet d'inscrire le programme même en mémoire et de ne le composer que d'une suite d'adresses de cases, le programmeur prenant soin de déposer dans ces cases les codes des opérations à exécuter à chaque pas et des données à traiter. Vous avez constaté que cette *valse parmi les adresses* a donné la clé de l'ordinateur et montré qu'il se compose de trois grandes parties : une fabrique qui effectue les opérations mêmes et s'appelle l'*unité de traitement;* une *mémoire* qui enregistre la suite des mots logiques formant le programme ainsi que les mots numériques exprimant les données ou les résultats intermédiaires; un secrétariat appelé *unité de contrôle* qui interprète les mots logiques du programme et prépare les circuits de l'unité de traitement à bien exécuter les besognes qu'on leur assigne.

2

L'art des programmes

Dans ce chapitre nous allons scruter plus avant l'art de la programmation, en exposant d'abord ses voies et moyens en langage *homo*. Ce n'est qu'au chapitre suivant que nous verrons comment des programmes conçus par l'homme sont transcrits en langage *ordino* et deviennent ces espèces de « rôles de théâtre » que nous avons évoqués, ne contenant que des adresses qu'on appelle en scène tour à tour.

Maman prépare un gâteau feuilleté

Quand, au cours des chapitres précédents, vous avez compris ce qu'était un programme d'ordinateur, vous avez peut-être conclu que vous saviez désormais sur ce point tout ce qu'il importait que vous sachiez. Vous étiez loin du compte. Vous n'avez eu une vision nette que des tout premiers rudiments du sujet, les programmes courts et d'un seul jet, quelque chose comme les protozoaires de la programmation. En réalité, la faune des programmes comprend des espèces infiniment plus complexes, où des lignes d'action de longueurs variées se croisent et se conditionnent, s'enlacent et se délacent selon des cheminements qui peuvent être extraordinairement touffus. Pour vous convaincre qu'il y a plus de choses dans cette faune qu'on ne le soupçonne au premier regard, examinons un cas encore très simple, mais qui cependant fait naître d'emblée l'une des complications majeures, qu'on appelle le « pas conditionnel ».

Maman veut préparer un gâteau feuilleté en utilisant sa nouvelle pétrisseuse mécanique qui comporte à l'entrée un plateau E où l'on dépose la pâte, puis la machine proprement dite qui absorbe le contenu du plateau E et le pétrit de plus en plus finement si on y insère des rouleaux R_1, R_2, R_3, ... de plus en plus fins. Enfin, à la sortie, un second plateau S qui recueille la pâte pétrie (*). Pour étrenner glorieusement sa machine, Maman voudrait réussir le gâteau à la perfection, selon l'exacte recette qui recommande d'arrêter le pétrissage lorsque la pâte est devenue une feuille d'épaisseur $e = 0{,}2$ mm. Elle n'a pas l'œil d'un géomètre et voudrait en outre élaborer un programme d'action que quelqu'un d'autre pourrait exécuter au cas où elle serait appelée ailleurs par les cris de bébé.

(*) Inutile de dire que cette pétrisseuse à rouleaux variés est un gadget purement imaginaire qui ne répond à rien d'autre qu'à me fournir une illustration.

Les débuts du programme sont évidents. On pose sur le plateau d'entrée E la première pâte (achetée pré-pétrie chez la boulangère moderne), on insère le premier rouleau R_1, on fait tourner la machine, et on recueille le produit dans le plateau de sortie S. Et dès cette fin de la première séquence de pas, Maman devient perplexe : ce produit est-il déjà bon pour la suite, ou faut-il recommencer un nouveau pas de pétrissage ?

Vous savez bien comment répondre à ce problème : il faut, à chaque fin de pétrissage, mesurer l'épaisseur p de la pâte qui sort, et la comparer à la valeur e imposée par le livre de cuisine : si p est plus grand que e, il faudra faire une nouvelle passe de pétrissage avec le rouleau suivant; si p est plus petit que e (ou égal à e), on doit arrêter l'opération, et envoyer son produit au beurrage, ou au four, ou à telle autre phase de la préparation prévue par la recette. Vous voyez qu'ici nous ne pouvions pas *d'avance* donner une suite d'ordres se déroulant d'*un seul jet* : car au beau milieu de l'action, il faut procéder à un **test,** et selon le résultat de ce test, qu'on ne peut pas prévoir, il faudra aiguiller la suite de l'action sur telle voie ou telle autre. On dit qu'en ce pas se situe un « **branchement conditionnel** ».

Observez soigneusement, je vous prie, ce qui est connu et ce qui n'est pas connu d'avance dans un tel pas. Ce qui est connu d'avance, c'est d'abord qu'il faudra, arrivé en ce pas, procéder à un test, et on sait quel test. On connaît aussi d'avance *toutes les voies d'action qui s'ouvrent au delà*, à raison d'une voie pour chacun des résultats *possibles* du test. Mais ce qu'on ignore, c'est quel résultat le test va donner. C'est bien pourquoi on doit faire entrer dans le programme toutes les voies, donc toutes les éventualités possibles, sous une forme telle que la suivante :

$$\begin{cases} \text{si } p > e, \text{ revenir à tel pas;} \\ \text{si } p \leqslant e, \text{ aller à tel pas;} \end{cases}$$

où le caractère conditionnel saute aux yeux.

Les actes ou événements conditionnels occupent une position stratégique dans toutes les entreprises de l'homme et toutes les manifestations de la nature. Lorsque je tends le bras pour saisir un objet, un mécanisme biologique dont je ne suis pas conscient mesure, à chaque position de mon bras, la distance qui sépare ma main de l'objet. Si elle est plus grande qu'une certaine valeur, un ordre intérieur instruit mon bras de continuer à se tendre. Si elle est plus petite, un autre ordre instruit ma main de s'ouvrir pour se préparer à agripper. L'éducation de notre système sensori-moteur consiste essentiellement à rôder et mettre au point ces mécanismes d'auto-contrôle qui font de tous nos gestes des réalisations de programmes parsemés de branchements conditionnés par des comparaisons. Toute la vie animale et végétale dépend de mécanismes régulateurs de ce genre, toujours basés sur le choix d'une voie ou d'une autre selon le résultat non prévisible d'un test. Ils commandent aussi l'attitude de nos banquiers : si le montant d'un chèque signé par nous est approvisionné, ils payent; sinon, ils appellent le commissariat de police. Donc le programme par lequel on pourra rendre automatique l'action du caissier qui se voit présenter un chèque contiendra, en son principal carrefour, un branchement conditionnel.

Nous avons aussi rencontré de tels branchements dans le programme du bon marieur. Rappelez-vous : si elle est blonde, me proposer la femme quelle que soit sa fortune, sinon, ne me la proposer que si elle est rousse et plus riche que tant. Dans cet exemple on avait même une condition seconde emboîtée dans une condition

première. La tenue d'un stock est aussi gouvernée à chaque pas par des tests : avant de savoir si on rachète ou non tel article, il faut examiner s'il en reste ou non en stock une quantité suffisante. Ce que j'ai appelé plus haut la méthode des approximations successives pour résoudre un problème mathématique fournit un autre exemple : à la fin de chaque cycle, il faut voir si l'erreur qui reste est plus petite ou plus grande qu'une tolérance donnée; si elle est plus grande, on réitère un nouveau cycle, si elle est plus petite on arrête et on prend le dernier résultat obtenu comme bon. Vous voyez que les branchements conditionnés pénètrent toutes les phases de la vie et de nos activités. Un ordinateur qui ne saurait pas les traiter, et n'accepterait que des programmes d'un seul jet, serait complètement inepte.

L'ordre de branchement

Comment un ordinateur accommode-t-il des ordres de branchement conditionnel? Pour bien le comprendre, il faut distinguer, dans un tel ordre, deux parties. La première partie, servant de préambule, est le *test*, *c'est-à-dire la comparaison* entre deux nombres qui, dans l'exemple précédent s'appelaient p et e et que, d'une façon générale, j'appellerai r et s. Cette comparaison peut avoir l'un des trois résultats suivants :

$$1 : r > s, \qquad 2 : r = s, \qquad 3 : r < s.$$

Cette partie devra donc être exécutable par un ordre spécial de comparaison. La seconde partie constitue l'ordre de branchement proprement dit, mais muni de l'un des trois indices 1, 2, 3 précédents. Notons cet ordre « BRAN » (par abréviation de « branchement ») suivi de l'indice 1, ou 2, ou 3. Dans le déroulement des pas d'un programme, au moment où j'en arriverai à un branchement conditionnel dont la suite dépend de la relation régnant, en ce pas, entre deux nombres r et s, j'écris, par exemple, le mot logique

$$\text{BRAN } 3 \rightarrow P.$$

Cela aura pour effet que si la comparaison entre r et s donne effectivement le résultat 3 (c'est-à-dire $r < s$) la machine va *sauter* (*se brancher*), non au pas suivant, mais à tel pas ultérieur que j'aurai mis dans le vade-mecum sous le N° P. Attention, attention! pour la première (et seule) fois, nous rencontrons ici un mot logique où le nombre inscrit après l'ordre ne représente *pas* une adresse.

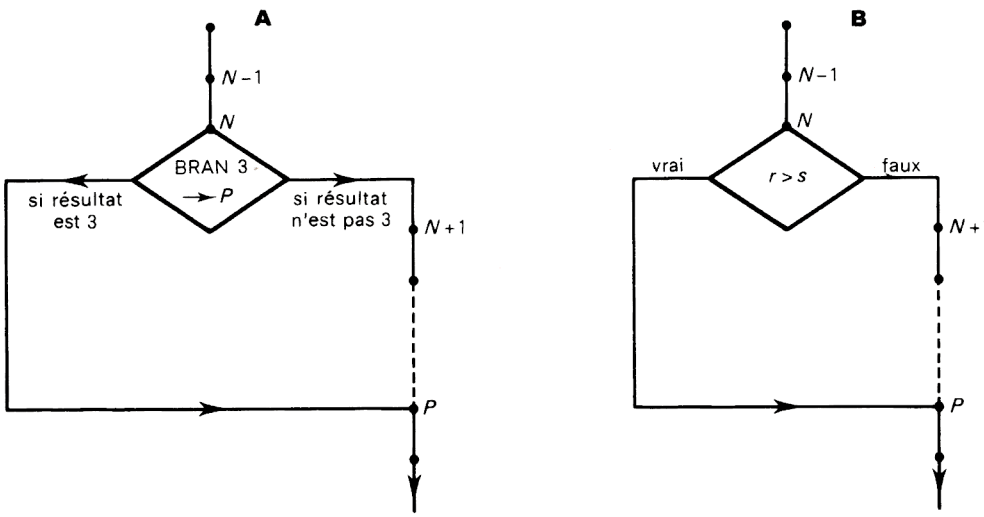
▶ — Que représente-t-il, s'il vous plaît?

▶▶ — Le numéro du *pas de programme* auquel on saute si le test réussit, c'est-à-dire fournit le résultat que vous avez cité après les lettres « BRAN ».

▶ — Et si le test fournit l'un des deux *autres* résultats possibles, 1 ou 2, c'est-à-dire ceux sur lesquels on n'a *pas* misé, alors l'effet de l'ordre BRAN $3 \rightarrow P$ sera de brancher le vade-mecum, *automatiquement*, sur le pas de programme *adjacent* à celui où s'était placé le branchement conditionnel.

La figure **A** illustre ce mécanisme. Au lieu d'utiliser un indice (1, 2 ou 3) après le mot « BRAN », on peut aussi écrire directement la condition de test dont l'observance

ou l'inobservance branche le programme sur une voie ou une autre, et appeler les sorties vers ces deux voies « vrai » ou « faux », selon la figure **B.**



Voyons ce que tout ça donne dans l'affaire de maman gâteau.

Son programme commence, au premier pas, par un ordre que j'écris « GARN 10 ». Qu'est-ce que cela veut dire? « GARN » est l'abréviation de « garnir » et instruit le système de garnir le plateau d'entrée *E* de la pétrisseuse avec la pâte achetée. Pour jouer à la poupée *ordino*, j'ai désigné *E* par le nombre 10 comme si c'était l'adresse d'un casier dans une mémoire *ordino*. Donc nous avons d'abord l'ordre « GARN 10 », à la suite duquel la pâte se trouve sur *E*. Le second ordre va être d'instruire la machine de pétrir avec le rouleau N° 1. Supposons que les rouleaux de finesses croissantes sont déposés dans les casiers d'adresses 21, 22, 23 et la suite. Nous aurons donc d'abord un ordre d'aller chercher le rouleau le plus gros à l'adresse qu'il faut, ordre que nous allons écrire « CHER 21 ». C'est le second pas du programme. Le rouleau convenable étant maintenant dans la machine, il suffit, au troisième pas, de donner l'ordre « PETR » de pétrir, avec le rouleau contenu à l'adresse 21. On n'a pas besoin de préciser l'adresse de ce que la machine pétrira, car elle le fera automatiquement avec la pâte qu'elle trouve en *E*. Et, automatiquement aussi, le résultat se dépose dans le plateau de sortie *S*. C'est maintenant que se produit le test suivi de l'ordre de branchement. Le quatrième ordre sera donc de comparer l'épaisseur *p* de la pâte située dans *S* au standard *e* du livre de cuisine.

Écrivons cet ordre : « COMP (*p*, *e*) » (Je vais ici un peu plus vite que la musique. En fait, nous savons que l'*ordino* ne peut marcher que d'adresse en adresse. Il faut donc avoir préalablement déposé la valeur de *E* dans un casier *U* de la mémoire, et déposer dans un autre casier *V* la valeur qu'on vient de trouver de *p*. L'ordre de test s'écrira alors, plus correctement « COMP (*u*, *v*) » voulant dire « comparer le contenu de l'adresse *U* avec le contenu de l'adresse *V* ». Je court-circuite ce petit cérémonial en écrivant d'emblée « COMP (*p*, *e*) ».)

Le résultat de ce 4e ordre est, ou bien 1, $p > e$, ou bien 2, $p = e$, ou enfin 3, $p < e$. Je *mise sur 3* et donne donc pour 5e ordre « BRAN 3 → 7 ». Cela veut dire que, si la comparaison donne bien le résultat 3, c'est-à-dire $p > e$, ce qui exige la réitération du pétrissage, je vais *sauter* au 7e pas du programme. Pourquoi au 7e, et pas au 6e, qui est le pas adjacent? Parce que là je vais inscrire ce qu'il faut faire si j'ai *mal* misé, c'est-à-dire si p s'est avéré *inférieur* à e; auquel cas, comme je vous l'ai dit, je dois non pas répéter le cycle, mais continuer par une autre voie. Quelle est cette voie? C'est clair : si $p < e$, il faut terminer le pétrissage. Donc l'ordre occupant le 6e pas sera : « EXTR S » qui veut dire ... je vous le laisse deviner.

▶▶ — Extraire le contenu du plateau S.

▶ — Si, au contraire, j'ai bien misé, c'est-à-dire si p s'avère encore, à ce pas, supérieur à e, je suis envoyé au 7e pas pour recommencer un cycle. Vous vous demandez sans doute pourquoi j'ordonne ce saut à un 7e pas au lieu d'un saut vers le retour pur et simple du cycle de pétrissage? C'est parce que, avant ce retour pur et simple, il faut encore un petit préparatif, vu que le nouveau cycle ne sera pas rigoureusement identique au premier sur un point. Voyez-vous lequel?

▶▶ — D'après votre drôle d'invention, il faut changer de rouleau.

▶ — En effet. Donc l'ordre initiant un second cycle, qui va remplacer l'ordre N° 2 précédent et auquel on doit retourner, devra être « CHER 22 » et non plus « CHER 21 ». A cet effet, j'exécute au pas N° 7, un ordre spécial qu'on appelle « Modification d'adresse », et que je vais écrire en abrégé « MOAD ». C'est un ordre qui a pour effet de modifier une adresse selon une certaine loi. Pour bien comprendre ce petit jeu, désignons l'adresse à modifier non pas par sa valeur numérique, mais par une lettre générique telle que (n). Dans notre cas, puisqu'on a déposé les rouleaux de finesses croissantes 1, 2, 3, ... dans les casiers d'adresses 21, 22, 23, ... il faudra, si n désigne l'une quelconque de ces adresses, qu'on fasse d'abord $n = 21$, puis qu'à chaque fois on augmente n de 1. L'ordre complet de modifications s'écrira donc « MOAD $(n, +1)$ » et ce sera notre 7e pas de programme. Notez qu'en même temps l'ordre N° 2 ne doit plus s'écrire « CHER 21 » mais « CHER n » puisque ce pas N° 2 va devenir désormais le début d'un cycle *quelconque*. Il faut toutefois que le programmeur n'oublie pas d' « initialiser » les cycles en faisant une première fois $n = 21$.

Ce préparatif étant achevé au 7e pas de programme, il faut effectivement effectuer le saut de retour vers le 2e pas. Quel ordre allons-nous employer à cet effet?

On pourrait songer à créer un ordre spécial de *saut*. Mais il y a mieux à faire si on réfléchit que l'ordre « BRAN (1 *ou* 2 *ou* 3) → P » dont nous avons fait connaissance tout à l'heure, prescrit déjà un saut. Rappelez-vous que le nombre P, dans ce mot logique, n'est pas une adresse mais un numéro d'ordre de pas de programme, et le mot instruit en effet d'interrompre le déroulement normal du programme pour sauter à son P-ième pas. Plus exactement, un ordre tel que « BRAN 3 → P » prescrit de faire ce saut si le test qui précède a réalisé le résultat 3 sur lequel on a misé. Cela en fait ce qu'on appelle un « **saut conditionnel** ». La seule différence avec le saut que nous voulons faire maintenant, celui qui nous ramènera du 7e pas au 2e, est que ce dernier saut doit *toujours* se faire, c'est-à-dire ne dépendra d'aucun test. On exprime ce fait en disant que c'est un « **saut inconditionnel** ». Qu'à cela ne tienne. Il suffit d'ajouter après l'ordre « BRAN » une quatrième éventualité, qu'on appellera par exemple « t », qui aura pour effet de brancher le programme sur le pas qu'on veut *dans tous les cas*, sans

regarder si un test antérieur a fourni un résultat 1, ou 2, ou 3. Nous allons bien sûr écrire au 3e pas « BRAN $t \rightarrow 2$ », ce qui nous renverra donc inconditionnellement au 2e ordre, et tout le cycle de pétrissage recommence automatiquement, et avec le nouveau rouleau.

Nous sommes maintenant outillés pour écrire — toujours en *homo*! — le programme du pétrissage automatique de la pâte par la machine de maman. Le voici.

Nos des pas de programme	Ordre exécuté a chaque pas	Effet
1	GARN 10	la pâte est placée à l'adresse 10 (= plateau d'entrée E);
2	CHER n	on va chercher le rouleau à l'adresse n; initialisation avec $n = 21$;
3	PETR n	la machine pétrit avec le rouleau contenu à l'adresse n;
4	COMP (p, e)	on compare l'épaisseur p de la pâte obtenue à la valeur e;
		$1 : p < e$; $2 : p = e$; $3 : p > e$;
5	BRAN $3 \rightarrow 7$	si 3 se réalise, on saute au pas distant 7; sinon on continue au pas adjacent 6;
6	EXTR S	le contenu du plateau S est extrait de la machine;
7	MOAD $(n, +1)$	l'adresse désignée par n augmente de 1;
8	BRAN $t \rightarrow 2$	on saute inconditionnellement à l'ordre No 2, mais la valeur de n qui s'y trouvait vaut maintenant $n + 1$.

Vous venez, cher lecteur, d'écrire votre premier programme. Comme c'est sous ma dictée, et que moi-même me suis inspiré de la science *ordino* existante, les devoirs de la modestie ne nous interdisent, ni à vous, ni à moi, d'admirer la belle économie de moyens que nous avons réalisée. Avec 8 ordres seulement nous avons obtenu que la machine répète autant de fois qu'il faudra — 10, 100, 1 000 fois — un cycle identique mais où une donnée change chaque fois (ici la finesse du rouleau). Et nous avons aussi introduit un mécanisme pour « sortir » du cycle, par un test qui arrête automatiquement l'itération et extrait le résultat final quand il répond à la norme que nous nous étions fixée. Et tout ça marche tout seul. Maman peut aller moucher bébé, la pâte sortira infailliblement au moment précis où elle sera parfaite.

Vous vous organisez pour extraire une racine carrée

Le mécanisme des « boucles de programme » que nous venons d'examiner, dont l'itération démarre ou s'arrête par des sauts conditionnels ou inconditionnels dans le vade-mecum, est si fondamental que nous allons le reprendre à propos d'un exemple plus proche des calculs qu'exécute un ordinateur. En outre, cet exemple va

nous révéler d'autres petits secrets de la valse des adresses. Il contient un peu de mathématiques, je m'en excuse, mais si peu que vous les maîtriserez sûrement sans peine.

Il s'agit d'extraire la racine carrée $Z = \sqrt{x}$ d'un nombre donné x positif. Vous savez que cette opération exige tout un petit algorithme. Celui qu'on emploie quasiment toujours dans l'univers *ordino* consiste à s'approcher de $\sqrt{x}$ par *approximations successives*. Supposons qu'au n-ième pas de ce processus, on ait obtenu, pour estimation approchée de $\sqrt{x}$, une certaine valeur, que je vais noter Z_n, le rang de ce pas étant comme vous voyez écrit en *indice*. Donc la valeur approchée que je vais obtenir au pas suivant se notera ... comment s'il vous plaît?

▶▶ — Sans doute Z_{n+1}.

▶ — Évidemment. Eh bien, un algorithme d'approximations successives est défini lorsqu'on donne une règle permettant de calculer Z_{n+1} en fonction de Z_n, ce qu'on appelle une « règle de récurrence ». Considérez par exemple, la règle de récurrence suivante :

$$Z_{n+1} = \frac{1}{2}\left[Z_n + \frac{x}{Z_n}\right].$$

▶ — Pourriez-vous écrire la valeur Z_{n+2}, celle qui suit d'un cran Z_{n+1}, en fonction de Z_{n+1}?

▶▶ — Bien sûr. Je change simplement n en $n+1$ dans votre formule et j'obtiens

$$Z_{n+2} = \frac{1}{2}\left[Z_{n+1} + \frac{x}{Z_{n+1}}\right].$$

▶ — Excellent. Vous avez fort bien saisi le mécanisme de la récurrence. Eh bien, Newton avait démontré que si on construit les valeurs successives Z_1, Z_2, Z_3, Z_4, ... Z_n, Z_{n+1}, Z_{n+2}, ... selon la règle précédente, plus on va loin, plus la valeur s'approche de Z, la racine carrée cherchée de x.

▶▶ — Permettez que je vous interrompe. Je vois bien comment on passe d'une valeur à la suivante, mais vous ne m'avez pas dit comment on choisit la toute première, c'est-à-dire Z_1.

▶ — Eh bien, on prend pour Z_1 simplement le nombre x lui-même. Faisons ensemble un essai dans le cas où $x = 4$. Nous savons qu'il faudra trouver $z = 2$. Partez d'une première valeur $Z_1 = x = 4$, et écrivez les trois prochaines valeurs Z_2, Z_3 et Z_4.

▶▶ — C'est facile. On a :

$$Z_2 = \frac{1}{2}\left[Z_1 + \frac{x}{Z_1}\right] = \frac{1}{2}\left[4 + \frac{4}{4}\right] = \frac{1}{2}[4+1] = \frac{1}{2}\cdot 5 = 2{,}5;$$

$$Z_3 = \frac{1}{2}\left[Z_2 + \frac{x}{Z_2}\right] = \frac{1}{2}\left[2{,}5 + \frac{4}{2{,}5}\right] = \frac{1}{2}[2{,}5+1{,}6] = \frac{1}{2}\cdot 4{,}1 = 2{,}05;$$

$$Z_4 = \frac{1}{2}\left[Z_3 + \frac{x}{Z_3}\right] = \frac{1}{2}\left[2{,}05 + \frac{4}{2{,}05}\right] = \frac{1}{2}[2{,}05+1{,}951] = \frac{1}{2}\cdot 4{,}001 = 2{,}000\,5.$$

▶ — Vous voyez que l'algorithme « converge », comme on dit, rapidement, puisque dès le 4e pas la valeur obtenue ne diffère de la valeur théorique exacte, qui est 2, que de 5 dix-millièmes. Pour des valeurs plus grandes de x il faudra sans doute plus de pas, mais peu importe puisque l'ordinateur les franchira tous automatiquement jusqu'à ce que nous soyons contents de l'approximation réalisée.

▶▶ — Mais quand serons-nous contents ?

▶ — Il faut, bien entendu, se donner d'avance une *erreur* (ε) *qu'on admettra*, c'est-à-dire préciser qu'on cherche la valeur de $\sqrt{x}$ non pas idéale, qui est inaccessible, mais approchée de l'idéal à ε % près, par exemple à 0,01 % près. Cette tolérance fait partie du but que je me fixe.

▶▶ — Et comment saurai-je quand je suis parvenu au voisinage de $\sqrt{x}$ à ε % près, puisque je ne sais pas combien vaut $\sqrt{x}$ idéalement ?

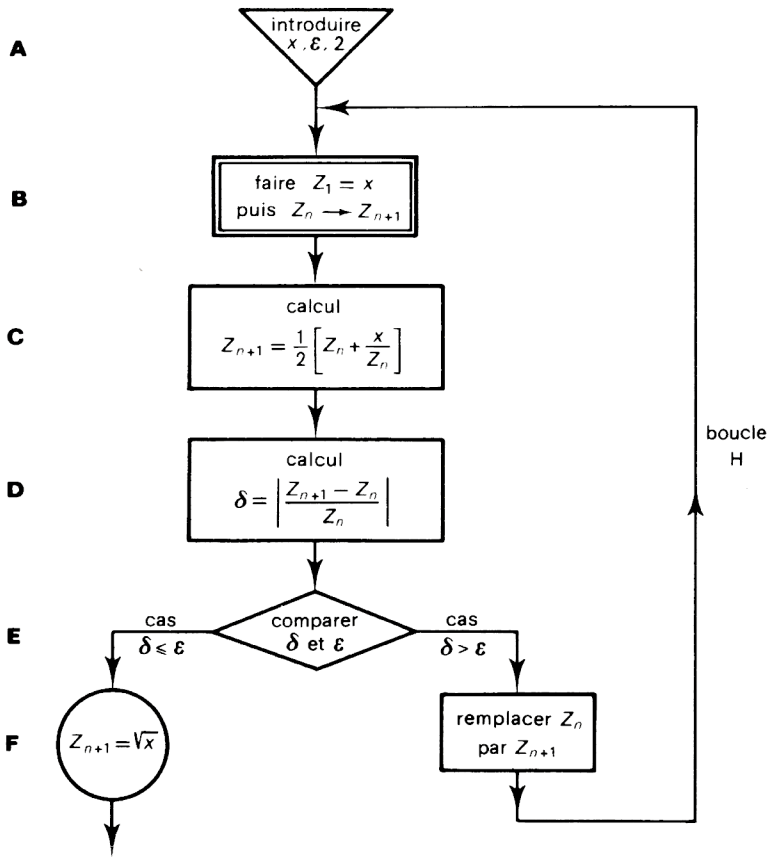
▶ — Excellente question. Puisque je ne peux pas comparer une valeur atteinte lors de tel ou tel pas à une référence, il me faut un autre critère pour décider si j'ai ou non pénétré dans la marge d'erreur admise autour de $\sqrt{x}$. Ce critère, Newton me l'a fourni en démontrant que, pour que je sois sûr d'avoir pénétré dans ladite marge, il faut et il suffit que la *différence* entre *deux valeurs successives*, rapportée à l'une d'elles, c'est-à-dire la quantité $\delta = \dfrac{Z_{n+1} - Z_n}{Z_n}$ soit devenue inférieure à ε en valeur absolue. Ce critère là est applicable puisqu'il travaille, non plus sur l'inaccessible $\sqrt{x}$, mais sur deux valeurs successives de ma récurrence, que j'ai calculées.

Dès lors, nous possédons un test pour achever les itérations : à chaque tour on calcule δ, et s'il s'avère supérieur à la tolérance admise, on recommence un tour; s'il s'avère égal ou inférieur, on sort comme bonne la valeur atteinte à ce tour. On pourra garantir qu'elle est égale à $\sqrt{x}$ à ε % près.

Vous voyez surgir à nouveau un branchement conditionnel. Cette fois nous allons tracer, avant d'écrire le programme, un schéma qui représente ses grandes articulations afin de soutenir notre pensée et guider notre marche. Un tel schéma s'appelle l'**organigramme** du programme, parce qu'il ressemble aux fameux diagrammes qui définissent les lignes directrices d'une organisation et qui commencent par le P.-D.G. pour descendre à travers une arborescence plus ou moins touffue d'échelons hiérarchiques jusqu'aux services du courrier et de la réception. La figure ci-contre représente cet organigramme. Ne vous laissez pas effrayer, entrez dedans carrément et suivez le guide, vous verrez que ça vous amusera.

La première phase, marquée **A**, consiste à faire entrer en mémoire les « constantes » dont on aura besoin durant le calcul : ce sont naturellement le nombre x lui-même dont on veut extraire la racine carrée, l'erreur relative ε qu'on admet, et aussi, ne l'oublions pas, le petit nombre 2, car c'est par lui qu'à chaque pas on devra diviser la quantité $Z_n + \dfrac{Z_n}{x}$. Le rectangle **B** à cadre double contient l'opération qu'on appelle « initialisation » du cycle. La toute première fois, cela consiste à prendre pour Z_1 la valeur x. Pour que l'ordinateur fasse cela, nous décidons de consacrer un casier déterminé de la mémoire à recevoir les nombres qui, au cours des cycles, joueront le rôle de Z_n, et nous ferons ensuite passer x dans ce casier spécialisé. Cela suffira

pour signifier à la machine qu'elle peut considérer, et doit traiter, x comme un Z_n, en l'espèce comme Z_1. Le premier cycle de calcul commence alors, et c'est ce que



représente le rectangle **C**. Dans ce premier cycle on fera donc l'opération $\dfrac{Z_1 + \dfrac{x}{Z_1}}{2}$. Par la suite, au n-ième cycle, quand le casier spécial débitera Z_n et non plus Z_1, ce sera l'opération $\dfrac{Z_n + \dfrac{x}{Z_n}}{2}$. Le résultat de cette opération est Z_2, la première fois, et par la suite, en général, ce sera Z_{n+1}. Il faut un autre casier spécial pour y déposer ce résultat Z_{n+1}, car on aura à faire un calcul avec lui, à savoir celui de la différence relative $\delta = \dfrac{|Z_{n+1} - Z_n|}{Z_n}$, calcul qui est l'objet de la phase du programme contenue dans le rectangle **D**. Si vous avez l'œil perçant, vous aurez noté que la différence

$Z_{n+1} - Z_n$ est entourée de deux barres verticales : cela veut dire qu'on en prend la *valeur absolue*, car son *signe* n'importe pas. Ayant formé δ, on entre en **E** dans le test — qu'il faut faire *à chaque pas*! — consistant à comparer δ avec ε. Vous savez déjà la suite : si on trouve $\delta \leqslant \varepsilon$, on sort en **F** la valeur trouvée en ce pas, et on l'adopte pour $\sqrt{x}$; si on trouve $\delta > \varepsilon$, on remplace Z_n par Z_{n+1}, ce qui se fait simplement en poussant le contenu du casier réservé aux Z_{n+1}, dans le casier réservé aux Z_n, ce qui est représenté par la flèche H de bouclage. La machine, trouvant quelque chose de nouveau dans ce dernier casier, opère dessus exactement comme elle l'avait fait auparavant; car, rappelez-vous le « plan von Neumann », **elle n'est sensible qu'aux adresses** *et exécutera toujours ce qu'on lui a ordonné de faire à partir d'une adresse, quel que soit le contenu de celle-ci.* Donc un nouveau cycle recommence avec des valeurs qui ont progressé d'un cran. Ce cycle suivra identiquement le même chemin que le précédent, *sans qu'on ait à écrire à nouveau toute la liste des ordres qui font emprunter ce chemin.* Vous voyez la puissance et l'élégance de la « marche par adresses ».

Le tracé des organigrammes est devenu tout un art. En particulier, on a standardisé les formes — triangles, rectangles, losanges, ronds, ovales — des enveloppes qui contiennent les divers grands types de traitement : insertion, calcul, comparaison, saut, extraction, etc. J'ai respecté ces conventions de dessin dans la figure précédente.

Vous programmez l'extraction de la racine carrée

Maintenant que nous avons tracé notre organigramme, nous allons écrire le programme correspondant (*). Nous postulons bien sûr que la machine dispose d'abord de tous les ordres arithmétiques nécessaires, d'addition (ADD), de division (DIV), de soustraction (SOU), etc. Vous voyez que j'abrège maintenant les ordres avec *3 lettres au lieu de 4*, car vous êtes devenu plus calé. En outre, la machine dispose des ordres logiques d'extraction (vers la sortie, EXT) de comparaison (pour le test, COM), de branchements inconditionnel (BRA *t*) et conditionnel (BRA 1, ou BRA 2, ou BRA 3) sur les éventualités

$$1\ (\delta < \varepsilon), \quad 2\ (\delta = \varepsilon) \quad \text{et} \quad 3\ (\delta > \varepsilon).$$

Mais je vais supposer d'autre part, que les mots logiques de cet ordinateur n'ont la place de contenir, en dehors d'un ordre, qu'*une seule* adresse (x, y).

▶▶ — Je ne vois là rien d'alarmant. C'est ainsi que vous m'avez présenté les mots logiques jusqu'ici.

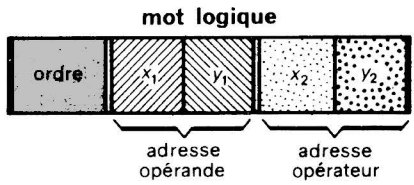
▶ — En effet, je ne voulais pas vous compliquer la vie. Il est temps maintenant d'approfondir un peu. Toute opération élémentaire porte sur *deux* quantités (ou objets, ou entités). Souvent même elles ne jouent pas des rôles symétriques et on distingue l'*opérande*, qui est la quantité passive sur laquelle on opère, et l'*opérateur* qui est la quantité active réglant l'opération faite sur la première. Cette répartition des rôles est parfois utile même quand les deux rôles sont en fait interchangeables,

(*) Les lecteurs pressés peuvent passer ce paragraphe.

c'est-à-dire quand l'opération est symétrique par rapport aux quantités opérées. Il en résulte que quand j'instruis l'ordinateur d'effectuer une opération, il faut qu'à la suite de l'ordre lui précisant quelle opération c'est, je lui donne *deux* adresses et non pas une seule, l'une indiquant où on doit aller chercher l'opérande, l'autre indiquant où on trouvera l'opérateur.

▶▶ — J'avais déjà vaguement senti qu'il y avait quelque insuffisance avec votre adresse unique.

▶ — Je n'en doute pas, tout le monde y pense. Et dans le premier âge des ordinateurs, leurs mots logiques contenaient en effet *deux* adresses, comme ceci :



▶▶ — Est-ce que ça ne conduisait pas à des mots bien longs?

▶ — Non, car à l'époque les ordinateurs avaient des mémoires exiguës, dont les adresses requéraient peu de digits. Remarquez que des ordinateurs « à deux adresses » (par mot logique) existent toujours. Mais la plupart des grosses machines modernes, ayant des mémoires considérables, ne portent qu'une seule adresse dans leurs mots logiques. Pour se débrouiller quand même, elles mettent en œuvre un **registre** spécial. On appelle « *registre* », en *ordino*, une filière d'alvéoles de mémoire ayant au moins autant d'alvéoles que les mots circulant dans la machine ont de places, et en général le double. Cette filière forme donc un grand casier, mais qui est situé à part, *en dehors de la mémoire*. Et nous verrons aussi que les alvéoles d'un registre n'ont pas tout à fait la même nature physique que les alvéoles de la mémoire. Un ordinateur possède en général plusieurs registres, accomplissant des rôles variés et répartis à divers endroits de la circuiterie, mais l'un d'eux est universel et fondamental, c'est celui qui est placé *immédiatement à la sortie de l'unité arithmétique* et qui recueille donc systématiquement le fruit des opérations qui s'y effectuent. Il possède en général une capacité double de celle des mots, afin de pouvoir accommoder les résultats des multiplications. Car vous savez que quand on multiplie deux nombres de n digits, on obtient un produit de $2n$ digits. *Après* l'avoir écrit, on peut « arrondir » en négligeant les digits les moins significatifs, ou encore continuer un moment avec ce nombre à $2n$ places, en le logeant dans 2 casiers au lieu d'un, jusqu'à ce qu'on soit sûr qu'on peut revenir au format standard des nombres à n places. C'est ce qu'on appelle une tranche de calcul « à double précision ». Quoi qu'il en soit, ce registre spécial appelé « **accumulateur** » et placé à la sortie de l'unité arithmétique, joue le rôle essentiel *d'héberger toujours l'opérande de l'opération à faire*. De cette façon, on n'a plus besoin de spécifier l'adresse dudit opérande. Un mot logique tel que « ADD 2850 » (j'écris tout en *homo*, donc l'adresse en décimal) signifiera qu'il faut *additionner le contenu de l'adresse 2850 au contenu de l'accumulateur*. Tel est le secret des ordinateurs modernes « à une adresse ».

Deux ordres spéciaux régissent les déplacements entre l'accumulateur et la mémoire. L'ordre « garnir », en abrégé « GAR », suivi d'une adresse Z, a pour effet de transférer le contenu du casier de la mémoire portant cette adresse dans l'accumulateur. L'ordre « ranger », en abrégé « RAN », suivi d'une adresse Z, a pour effet de transférer le contenu de l'accumulateur dans le casier Z de la mémoire. Notez que les rôles de l'adresse dans ces deux ordres sont opposés : dans l'ordre « GAR Z » on va *depuis Z vers ailleurs*, tandis que dans l'ordre « RAN Z » on va *depuis ailleurs vers Z*. Notez aussi que la combinaison des deux ordres permet de *transférer* un mot d'un casier A de la mémoire vers un autre casier B de la mémoire en deux étapes : il suffit d'ordonner la succession des deux ordres « GAR A », qui envoie le mot contenu situé à l'adresse A dans l'accumulateur, puis « RAN B », qui envoie le mot situé dans l'accumulateur vers l'adresse B. Vous vérifiez que dans tout ça on ne cite pas nommément le registre accumulateur, *il est toujours sous-entendu*. Ce registre est une pièce maîtresse de tout ordinateur.

Ceci dit, commençons la rédaction du programme en nous inspirant de l'organigramme de tout à l'heure. L'étape préparatoire inscrite dans le triangle A ne fait pas partie du programme proprement dit, car le « chargement » de la mémoire avec les constantes s'effectue à la main à partir d'un clavier. Mettons qu'on ait réservé les adresses suivantes :

— 100 pour y loger le nombre x,

— 101 pour y loger la constante ε,

— 102 pour y loger la constante 2,

et ajoutons tout de suite qu'on réservera les casiers

— 103 pour y loger les Z_n,

— 104 pour y loger les Z_{n+1}.

On doit d'abord, selon le rectangle **B** de l'organigramme, initialiser les cycles récurrents en prenant pour première valeur Z_1 le nombre x lui-même. Que faut-il faire à cet effet ?

▶▶ — Placer x dans l'adresse 103 où viennent loger les Z_n successifs.

▶ — Et pour cela, comme x loge à l'adresse 100, il faut faire un transfert mémoire → mémoire de cette adresse vers l'adresse 103, ce qui s'effectue comme nous venons de le voir par la succession des deux ordres « GAR 100 » et « RAN 103 ». (Dans la suite, les opérations effectuées successivement sont énoncées de la façon suivante : numéro de l'ordre, nature de l'ordre, adresse concernée, effet de l'ordre.)

1) GAR 100 : transporte x de mémoire 100 dans l'accumulateur.

2) RAN 103 : transporte x de l'accumulateur dans mémoire 103.

Maintenant va commencer l'opération indiquée dans le rectangle **C** de l'organigramme, c'est-à-dire le calcul $\dfrac{Z_n + \dfrac{x}{Z_n}}{2}$ qui, la première fois, s'écrit $\dfrac{Z_1 + \dfrac{x}{Z_1}}{2}$.

Que, dans ce premier tour d'itération, la valeur Z_1 traitée, celle qui occupe l'adresse 103, soit x lui-même, ne doit pas nous retenir d'ordonner cette opération en termes généraux, puisque, aux pas ultérieurs, on devra traiter de même, non pas $Z_1 = x$, mais Z_2, puis Z_3, puis Z_4, etc. Or, en général, il faudra, pour former $\frac{x}{Z_n}$, diviser x par le contenu de 103. Donc un premier travail doit être d'envoyer x, qui pour cette division jouera le rôle d'opérande, dans l'accumulateur, afin qu'on puisse ordonner ensuite « diviser par le contenu de 103 » sans plus avoir à mentionner ce qu'on divise, puisque se sera toujours, en sous-entendu, le contenu de l'accumulateur. La première fois, ce renvoi de x dans l'accumulateur au moyen de l'ordre :

3) GAR 100 : transmet le contenu de 100 (qui est x) dans l'accumulateur.

Cette instruction est un coup pour rien, puisque x a déjà siégé dans l'accumulateur au 1er pas du programme; mais ce coup est nécessaire pour rendre uniforme le parcours d'un cycle les fois suivantes. On peut maintenant écrire l'ordre :

4) DIV 103 : le contenu de l'accumulateur est divisé par le contenu de l'adresse 103.

Ici, attention, tendez votre esprit. La division précédente ayant été faite dans l'unité arithmétique, son résultat se retrouve *automatiquement*... où?

▶▶ — Dans l'accumulateur.

▶ — Dans l'accumulateur, où ce résultat aura chassé ce qui y siégeait auparavant. On n'a donc pas besoin de consacrer un ordre spécial à mettre le résultat d'une opération arithmétique dans l'accumulateur, il y tombe tout seul. C'est l'un des avantages principaux de ce fameux registre. Seul son garnissage à partir de la mémoire exige un ordre spécial.
Dès lors, quand, au pas suivant, il faut ajouter Z_n au quotient, ce dernier étant déjà dans l'accumulateur, il suffira d'écrire l'ordre :

5) ADD 103 : le contenu de 103, c'est-à-dire Z_n, est additionné au contenu de l'accumulateur.

Comme, ici encore, le résultat de l'addition, c'est-à-dire le nombre $Z_n + \frac{x}{Z_n}$ se trouve spontanément dans l'accumulateur, il suffit pour le diviser par le nombre 2, lequel loge à l'adresse 102, d'écrire l'ordre :

6) DIV 102 : on obtient le nombre $\frac{Z_n + \frac{x}{Z_n}}{2}$ siégeant dans l'accumulateur.

C'est ce nombre qui, par la règle de récurrence de Newton, doit devenir Z_{n+1}, la valeur au prochain tour d'itération. Or, pour l'ordinateur, *est* Z_{n+1}, *tout ce qui loge à l'adresse 104.* Donc, pour que le nombre $\frac{Z_n + \frac{x}{Z_n}}{2}$, qui siège dans l'accumulateur, devienne Z_{n+1}, il me suffit de l'envoyer dans l'adresse 104 de la mémoire, ce qui se fait par l'ordre :

7) RAN 104 : envoie $\frac{1}{2}\left(Z_n + \frac{x}{Z_n}\right)$ de l'accumulateur dans l'adresse 104 de la mémoire, ce qui en fait Z_{n+1}.

Vous observerez ici jusqu'à quelle profondeur va la méthode de la « marche par adresses » selon le « plan von Neumann » : *l'adresse signifie son contenu*! C'est par l'endroit où siège un mot qu'il va jouer tel rôle ou tel autre. Ayant donc mon Z_{n+1} à l'adresse 104, je pourrai commencer à préparer mon test en faisant le calcul inscrit dans le rectangle **D**. Il faut former d'abord la différence entre Z_{n+1} et Z_n. Cette fois les *deux* quantités de mon opération sont en mémoire. Qu'à cela ne tienne, j'envoie d'abord l'un d'eux, mettons Z_n, qui siège en 103, dans l'accumulateur par un ordre « GAR », puis j'écris « SOU 104 » qui a pour effet de soustraire de l'occupant de l'adresse 104, qui est Z_{n+1}, l'occupant de l'accumulateur, qui est Z_n d'après la manœuvre précédente. (Pour aller vite, je suppose que l'opération désignée par « SOU » forme aussi la valeur absolue de la différence.) Ayant la différence $|Z_{n+1} - Z_n|$, spontanément dans l'accumulateur, je la diviserai par Z_n, qui loge à l'adresse 103, simplement en écrivant l'ordre « DIV 103 ». D'où la séquence :

8) GAR 103 : Z_n entre dans l'accumulateur.

9) SOU 104 : Le contenu de l'accumulateur, c'est-à-dire Z_n, est soustrait du contenu de l'adresse 104, c'est-à-dire Z_{n+1}, et la quantité $|Z_{n+1} - Z_n|$ tombe dans l'accumulateur.

10) DIV 103 : le contenu de l'accumulateur, c'est-à-dire $|Z_{n+1} - Z_n|$, est divisé par le contenu de 103, ce qui fournit δ dans l'accumulateur, et les tâches du rectangle **D** sont accomplies.

Je suis prêt maintenant pour le test. Il faut d'abord comparer le nombre δ qui loge dans l'accumulateur au nombre ε que j'avais logé à l'adresse 101 (revoyez le début). Cela s'exécute par l'ordre :

11) COM 101 : le contenu de 101, c'est-à-dire ε, est comparé au contenu de l'accumulateur, c'est-à-dire δ fournissant l'un des trois résultats :

$$1\,(\delta < \varepsilon), \quad 2\,(\delta = \varepsilon) \quad \text{ou} \quad 3\,(\delta > \varepsilon).$$

Je mise sur le résultat 3 en commandant, au 12e pas, le branchement conditionnel « BRA 3 → 14 ». J'ai fait faire le saut jusqu'au pas distant 14 et non jusqu'au pas adjacent 13 parce que j'applique la norme d'après laquelle ce dernier pas doit prendre la main, c'est-à-dire que le programme continue son déroulement linéaire, lorsque le test a donné le résultat *contraire* à celui sur lequel on a misé. Ce pas adjacent, le N° 13, doit donc contenir l'ordre qu'il faut donner si δ s'est avéré inférieur ou égal à ε; et cet ordre, nous le savons d'après le rond **F** de l'organigramme, est d'extraire le Z_{n+1} qu'on vient de trouver pour le vendre au dehors comme $\sqrt{x}$. Comme Z_{n+1} loge dans 104, le 13e ordre sera « EXT 104 ». Et c'est bien au 14e pas que je saute si, lors du branchement, j'avais visé juste, c'est-à-dire si δ s'est avéré supérieur à ε. Que dois-je faire dans ce cas? Le rectangle **F** de l'organigramme me le dit : je dois recommencer un cycle en faisant jouer à Z_{n+1} le rôle que vient de jouer Z_n. Et comment vais-je obtenir cela? Simplement en renvoyant Z_{n+1}, qui siège à l'adresse 104, dans l'adresse 103 réservée à Z_n. Dès que Z_{n+1} est parvenu dans ce casier, il sera traité automatiquement comme le fut Z_n. Comme Z_{n+1} siège en ce moment dans la mémoire, (à l'adresse 104), son transfert à une autre adresse (103) devra emprunter le détour par l'accumulateur au moyen d'un ordre « GAR » suivi d'un ordre « RAN ». Quand Z_{n+1} siégera dans 103, un dernier ordre de saut *in*conditionnel devra faire reprendre le programme au 3e pas, là où démarre le cycle; ce sera l'ordre « BRA $t \rightarrow 3$ », qui réalise la « boucle » **H** de l'organigramme. On a donc la séquence suivante :

12) BRA 3 → 14 : si test réussit sur 3, c'est-à-dire $\delta > \varepsilon$, on saute au pas 14.

13) EXT 104 : si test ne donne pas le résultat 3, c'est-à-dire $\delta \leqslant \varepsilon$, on extrait Z_{n+1} comme $\sqrt{x}$.

14) GAR 104 : Z_{n+1} va de la mémoire (casier 104) dans l'accumulateur.

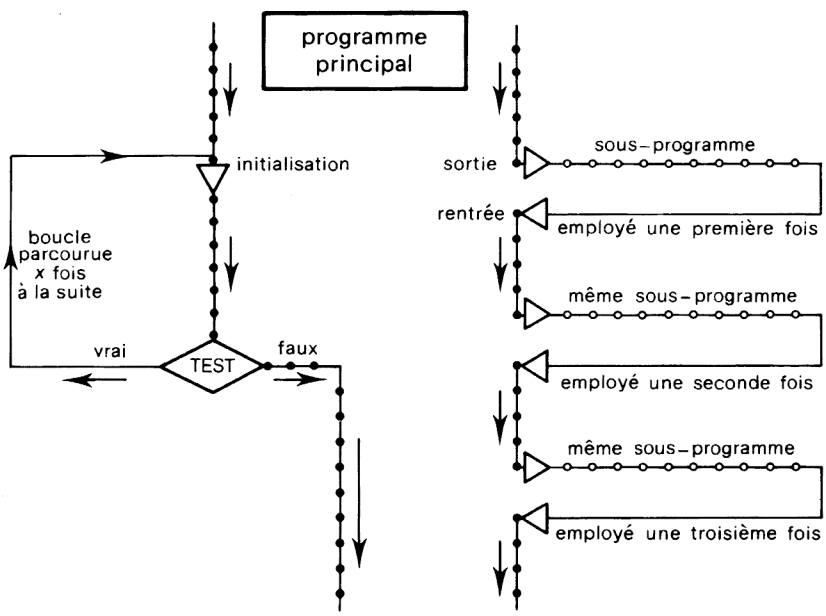
15) RAN 103 : Z_{n+1} va de l'accumulateur à la mémoire (casier 103).

16) BRA $t \rightarrow 3$: le programme saute (inconditionnellement) au pas N° 3.

Voilà notre affaire bouclée, c'est le cas de le dire. Peut-être avez-vous trouvé cette confection de programme un peu fastidieuse. J'ai tenu à vous le détailler parce que cela vous fait pénétrer non seulement dans les arcanes de la programmation, mais encore dans tout le mode de pensée qui anime l'informatique, fusion remarquable de pertinence et de rigueur dans le décorticage, de minutie et d'efficacité dans l'organigramme, d'élégance et d'astuce dans l'économie de mouvement. Si le premier panneau de la « révolution de l'informatique » est de pouvoir traiter des données dix millions de fois plus vite qu'à main d'homme, le second panneau est certainement d'avoir obligé l'homme à repenser profondément une multitude de ses activités pour les réduire à des séquences condensées d'actes simples sur des données traitables.

Les sous-programmes

Une « boucle » telle que celles que nous venons d'étudier s'insère dans le flot même du programme principal par une initialisation au début et un test à la fin. La boucle est parcourue un grand nombre de fois à la suite, jusqu'à ce que le test soit satisfait. Il existe un autre genre de « fragment de programme » qui s'inscrit aussi en « série » dans le programme principal, mais commence et se termine sur des ordres *in*conditionnels, et qu'on prépare d'avance non pas parce qu'il devra se dérouler cycliquement un grand nombre de fois à la suite, au même endroit du programme principal, mais parce qu'on prévoit qu'on en aura souvent besoin, à raison d'un parcours chaque fois, en divers endroits dudit programme. Ces fragments pré-cuisinés, qui sont en quelque sorte des *boucles ouvertes*, s'appellent des « **sous-programmes** ». La différence entre ce concept et celui de boucle est illustrée par la figure ci-dessous, assez clairement, j'espère, pour que je n'aie pas besoin d'y insister. Je préfère vous donner un exemple d'application des sous-programmes.



Vous savez, je suppose, ce qu'est une fonction mathématique. Je vous rappelle que c'est une règle, ou si vous préférez un « processus », par le jeu desquels une variable y est asservie à, ou rendue « *dépendante* » de, une variable x appelée « *indépendante* ». En ce sens qu'à toute valeur que peut prendre x *correspond* une valeur bien définie de y.

La règle, ou le processus, qui assurent la correspondance peuvent être :

— soit consignées dans une *table*, où on déploie les valeurs possibles de x dans une colonne et inscrit, en face de chacune d'elles, sur une colonne vis-à-vis, la valeur correspondance de y;

— soit incarnées par un *algorithme*, qui récite les opérations à faire sur une valeur de x pour engendrer la valeur correspondante de y.

Pour exprimer le fait que la variable dépendante y est liée à la variable indépendante x par une relation fonctionnelle dont on ne spécifie pas la nature, on emploie la notation

$$y = f(x)$$

qu'il faut lire « f ***de*** x », et où la lettre f mise devant les parenthèses est censée illustrer la règle, ou le processus, de la correspondance entre x et y. Si on précise ce processus, et qu'il dispose déjà d'une notation consacrée, c'est évidemment celle-là qu'on met à la place de la lettre banalisée f. Le cas le plus fréquent en mathématiques est celui où l'algorithme qui fait passer de x à y peut s'exprimer par une *formule*, par exemple

$$y = \frac{1}{\sqrt{x}}.$$

Cette fonction fait correspondre :

— à la valeur 4 de x, la valeur $\frac{1}{\sqrt{4}} = \frac{1}{2} = 0{,}5$ de y;

— à la valeur 9 de x, la valeur $\frac{1}{\sqrt{9}} = \frac{1}{3} = 0{,}333$ de y;

— à la valeur 16 de x, la valeur $\frac{1}{\sqrt{16}} = \frac{1}{4} = 0{,}25$ de y;

— à la valeur 25 de x, la valeur $\frac{1}{\sqrt{25}} = \frac{1}{5} = 0{,}2$ de y.

Vous sentez bien ici que si on obtient des valeurs différentes pour y, c'est parce qu'on part de valeurs différentes de x, mais la *loi* du passage de x à y *reste toujours la même*, c'est elle qui « incarne » *la* fonction.

On emploie en mathématiques, et dans les autres sciences, un très grand nombre de fonctions — les polynômes, le logarithme, l'exponentielle, le cosinus, le sinus, etc., et la racine carrée $\sqrt{x}$ qui nous a si longuement occupés, est de celles-là (vous ne butez pas, j'espère, sur le fait que plus haut j'ai appelé la variable dépendante Z alors que maintenant je l'appelle y. Les noms des variables sont affaire de convention pure).

Pour « *réaliser* » une fonction en une valeur donnée de sa variable indépendante, c'est-à-dire pour obtenir effectivement la valeur correspondante de sa variable dépendante, on doit, soit consulter une table, soit effectuer une série de pas de calculs, comme dans le cas de $\sqrt{x}$. Cette série de pas forme, nous l'avons vu, un *programme* et, chose capitale, un programme *qui est toujours le même quelle que soit la valeur de x à laquelle on l'applique*. C'est ce programme toujours identique à lui-même, prototype de ce qu'on appelle un « algorithme », qui incarne précisément *la* fonction dont on parle.

Que va-t-on faire, dès lors, si au cours d'un programme principal on a besoin de « réaliser » de nombreuses fois une même fonction f, à des endroits différents

pour des valeurs différentes de la variable indépendante, comme par exemple prendre le cosinus et le sinus de divers angles, ce qui arrive des dizaines ou centaines de fois dans des calculs d'arpentage, ou de bâtiment, ou d'astronautique? Va-t-on réécrire tout au long le programme P_f de la réalisation de f chaque fois qu'on en a besoin? Ce serait bien balourd. C'est là qu'apparaît l'idée de sous-programme. On écrit le programme P_f ***une seule fois***, et on l'introduit en mémoire, c'est-à-dire qu'on loge les instructions qu'il contient dans une suite de casiers numérotés depuis un numéro initial jusqu'à un numéro final. Puis, lorsqu'on arrive, dans le programme principal, à un pas N° n, à partir duquel on doit réaliser la fonction f avec une valeur donnée de x, on y insère un *ordre de sortie* vers le sous-programme P_f, comprenant aussi l'indication de la « valeur de départ » x sur laquelle il devra travailler. Cet ordre a pour effet d'aller vers le premier casier de la zone de mémoire où P_f a été logé, à partir de quoi on dévide les autres casiers de cette zone jusqu'à la fin de P_f. Lorsque cette fin est arrivée, on dispose de la valeur de y qui correspond à celle x de par la fonction f, et un « ordre de rentrée » la fait pénétrer dans le flot du programme principal au pas N° $n+1$ qui suit immédiatement le pas N° n où on était sorti.

Les sous-programmes sont également essentiels à la commodité de la programmation dans ce qu'on appelle les « **servitudes** ». Vous allez vous apercevoir peu à peu qu'un ordinateur est en réalité non pas *une* machine, mais un *ensemble* de machines. Cet ensemble est analogue à une réalisation d'urbanisme raffiné où plusieurs immeubles fonctionnels sont liés entre eux, chacun assumant un rôle auquel il est spécialement adapté. Vous verrez en particulier que l'unité centrale est entourée de ce qu'on appelle des *équipements périphériques* qui permettent à l'homme d'introduire dans, et d'extraire de, cette unité, les éléments qu'ils traitent. Certains de ces périphériques servent aussi, pendant le traitement même, de *mémoires auxiliaires*. Vous comprenez facilement que la circulation entre tous ces équipements exige des procédures parfois complexes, qu'on appelle des « *servitudes* » (par exemple, s'assurer qu'une voie est libre, qu'un registre est plein, distribuer les instructions de début et fin, garnir des zones spéciales de mémoire, délimiter des blocs, etc.). Ces servitudes devront naturellement être programmées, comme tout le reste dans le monde *ordino*. Mais elles se présentent très souvent à de nombreux carrefours de la plupart des programmes, et il serait inepte de les réécrire à chaque fois complètement. La solution est donc d'en faire des « sous-programmes ».

La monnaie intellectuelle de l'économie moderne

Les grands constructeurs d'ordinateurs écrivent d'avance les sous-programmes, les déposent sur des bandes ou paquets de cartes (nous verrons plus loin ce que cela signifie en pratique), donnent à chacun un nom ou titre (codé bien sûr) et vous les vendent en même temps que la machine pour que vous en formiez une « bibliothèque ». Cette bibliothèque constitue en somme le « *software de service* » de la machine. Elle apporte à l'usager une commodité d'emploi du plus haut prix, car elle le soulage de la nécessité d'écrire lui-même et indéfiniment des masses de programmes de routine, tout en le laissant libre d'insérer ce software de service où il le veut dans son programme principal propre. La fortune de la Compagnie IBM vient de ce qu'elle a

compris la première qu'il fallait vendre des bibliothèques entières de software de service en même temps que les machines.

Cette formule de *bibliothèque de software* a vite dépassé le cadre des sous-programmes servant aux servitudes ou au calcul des fonctions mathématiques les plus courantes. Elle s'est étendue à des *programmes entiers*, des programmes principaux et non plus accessoires, mais réalisés une fois pour toutes par des officines spécialisées et résolvant une foule de problèmes standard qu'on rencontre souvent dans la vie économique. Ces programmes entiers se vendent — fort cher — à tous ceux que leurs activités exposent à rencontrer souvent ces problèmes standard. La motivation dans ce cas est d'éviter à l'usager, non plus des routines, mais des migraines.

Il faut comprendre en effet que la confection des programmes est devenue un art qui, pour peu que les actions à programmer soient complexes, devient extraordinairement subtil et requiert de ceux qui le pratiquent une formation très poussée en même temps qu'une minutie surhumaine; puisque, s'il y a la plus petite erreur à un pas quelconque, tout le programme ne vaut rien. Songez seulement à tous les « trucs » de programmation que nous venons de passer en revue : les tests, les sauts conditionnels ou inconditionnels, les modificateurs d'adresse, les registres séparés, les itérations par boucles, les sous-programmes. Ils forment déjà une jolie collection, mais qui ne constitue encore que le ***B - A - BA*** du métier. On peut avoir des boucles dans les boucles, des boucles dans les sous-programmes, des sous-programmes dans les boucles, des sous-programmes qui consultent le programme principal, des transferts de blocs entiers de casiers à d'autres adresses, des transferts sélectifs d'adresses, (c'est-à-dire des transferts obéissant à une loi de tri, elle-même inscrite dans un sous-programme), des séquences conditionnelles dont la procession est ordonnancée par un sous-programme et des priorités, des itérations avec décalages savants, etc., sans compter les mille variantes pour l'entrée et la sortie des données. Quand il s'agit de problèmes un peu sérieux, le seul *organigramme* d'un programme peut présenter des entrelacs fantastiques de cheminements. Celui du programme qui régit la paie du personnel de la SNCF couvre cinq mètres d'un papier de 0,80 cm de large — tout à fait comme le plan d'une usine — et le programme lui-même qui réalise cet organigramme comprend quinze mille pas et a exigé plusieurs années de travail d'une équipe d'une douzaine de personnes.

A cette extrémité-là, on ne peut évidemment rien faire pour alléger la tâche de l'usager, d'autant que cette tâche lui est tout à fait particulière. Mais entre les problèmes de ce niveau et ceux de routine courante qui forment le « software de service », se placent beaucoup de problèmes de taille moyenne et de caractère universel, c'est-à-dire qui se rencontrent dans de nombreuses occasions différentes sous la même forme avec seulement d'autres paramètres. Ainsi le problème qu'on appelle en mathématique « l'inversion d'une matrice » se rencontre chez les économistes, les gestionnaires de stocks, les raffineurs de pétrole et les conquérants de la lune. Ce sont des problèmes de ce genre dont certains centres de calcul mettent les programmes entiers d'avance en conserve. Une boîte de ces conserves, contenant par exemple un programme tout fait d'inversion de matrice, ou d'interclassement de fiches, peut coûter quelques dizaines de milliers de francs. A mesure que la vie économique, scientifique et technique prend conscience des services qu'elle peut demander aux ordinateurs, on isole davantage de ces problèmes universels qui entrent dans beaucoup de mixtures. En conséquence, le commerce du software de confection s'enfle, et les

programmes standard d'ordinateurs tendent à devenir des éléments préfabriqués qu'on combine pour former l'outillage intellectuel de nos usines et bureaux.

▶▶ — La vision (me dites-vous, lecteur critique) est fort belle, mais elle me paraît se heurter à un problème majeur de compatibilité. Tels que vous me les avez décrits jusqu'ici, les programmes me paraissent intimement liés à des machines. Chaque machine n'a-t-elle pas un arsenal d'ordres qui lui sont propres, une mémoire différente de sa concurrente, une structure bien à elle des mots? Comment dès lors peut-on vendre un programme en conserve à des usagers qui possèdent des machines différentes?

▶ — Je suis bien aise que vous ayez posé cette question. Elle domine, en effet, le marché du software préfabriqué, et m'oblige à revenir d'abord sur la manière dont les programmes s'intègrent vraiment dans un ordinateur. J'allais oublier que je ne vous les ai présentés jusqu'ici qu'en langage *homo* et que je vous avais promis de vous dire après comment on les transpose en langage *ordino*. Mais nous allons traiter ces questions dans le chapitre suivant.

A la suite de maman qui s'organise pour qu'une machine prépare le gâteau automatiquement, on est entré un peu plus (un tout petit peu), dans les arcanes de la *programmation*, ou art de construire des programmes. On vous a montré la nécessité absolue de disposer d'ordres de *branchement conditionnel* qui envoient la poursuite des opérations à tel ou tel jalon de la séquence d'après les résultats d'un *test*. Après quoi, on vous a jugé assez avancé pour vous faire faire un petit programme. Ce qui vous a mis en conditions pour apprécier l'intérêt des *« boucles de programme »*, ou *« cycles d'itération »*, qui font recommencer identiquement la même suite de pas avec des données issues du cycle précédent, jusqu'à ce qu'un test signale qu'il y en a assez; et des *« sous-programmes »* qui sont des morceaux de programmes écrits d'avance pour réaliser des séquences se retrouvant dans beaucoup de problèmes. Vous avez constaté qu'à partir de là, on peut concevoir une vaste forêt de raffinements ayant tous pour but de permettre des écritures raccourcies capables de déclencher des séries importantes d'opérations imbriquées. Vous avez appris qu'il y a un commerce florissant de programmes et sous-programmes écrits d'avance et enregistrés.

3

Les langages

Le b - a - ba du début

Au début — et ceci se passait dans des temps très anciens, c'est-à-dire en 1950 — les usagers des ordinateurs écrivaient les programmes directement dans ce qu'on appelle le « ***langage machine*** » c'est-à-dire en tops et tips. Chacune des instructions que la machine était capable d'exécuter devait être désignée par le code binaire qu'elle portait, chaque adresse aussi, chaque donnée de même. Un programme achevé, sur le cahier du programmeur, avait donc l'aspect d'une longue liste de séquences de *zéros* et de *uns* écrites les unes au-dessous des autres.

Cette manière archaïque de programmer était abominable pour diverses raisons.

Elle était d'abord fastidieuse. On devait se rappeler par cœur le code binaire de tous les ordres, garder présent à l'esprit toutes les allées et venues qui avaient pu être proposées entre les diverses places de la mémoire afin que les mots ne se marchent pas sur les pieds, consulter constamment la numération binaire, décider des arrondis (pour respecter la taille des mots) au fur et à mesure et sans savoir quels seraient leurs effets sur la précision finale, etc.

Ensuite, le procédé était précaire. Il était aussi facile de commettre des erreurs — on a vite fait de remplacer un 0 par un 1 quand on en débite des centaines de mille — que difficile de les corriger. Car comment percevoir la substitution fautive par une simple relecture, surtout de « textes » aussi parfaitement rébarbatifs!

Enfin, la rigidité était extrême. Si on voulait modifier le chemin suivi à partir d'un point, on avait toutes les peines du monde à retrouver ce point. Il fallait aussi assigner d'avance des cases de mémoire à toutes choses, et si l'assignation décidée lors d'un pas s'avérait gênante, vingt pas plus loin il fallait tout découdre et recommencer.

Inutile de dire que les malheureux programmeurs de ces premières machines les maudissaient et trouvaient qu'elles étaient particulièrement bêtes, à ne savoir bouger que selon les interminables cadences de tops et tips, aveuglément, incapables de redresser les textes, c'est-à-dire faisant tout de travers si un seul des cent mille tops ou tips était faux. J'ai vécu personnellement cette époque et me rappelle combien on se gaussait, dans mon laboratoire, quand on voyait la presse appeler « cerveaux électroniques » ces réciteuses bornées de tops et moroses dévideuses de tips. Mais nos épreuves n'eurent qu'un temps, et ce fut encore von Neumann qui comprit comment on pouvait y mettre un terme en employant, pour écrire les programmes, un langage « symbolique » qui serait ensuite *interprété par un autre programme.*

Le principe des langages symboliques

L'idée est audacieuse, mais naturelle. Puisqu'un ordinateur est un esclave qui exécute sans rechigner tous les programmes que moi, son maître, je lui prescris, et puisque moi, son maître, je suis tellement plus subtil que lui, pourquoi m'imposerais-je la corvée de lui parler son assommant langage de hoquets? Je vais lui écrire mes ordres dans un langage formé de mots et symboles plus proches de mes habitudes à moi, et je fournirai d'autre part à mon esclave *un programme spécial pour interpréter mon programme symbolique*, c'est-à-dire pour transformer dans son naïf langage *ordino* le programme que j'ai écrit en *homo*. Peut-être aurai-je beaucoup de mal à concevoir ce *programme assurant la traduction de mes programmes homo*, mais peu importe, l'effort n'aura à être fourni qu'une seule fois! Car le programme de traduction emploiera, pour transposer ma sténographie en langage machine, des règles uniformes, valables pour *tous* les programmes symboliques de *tous* les problèmes. Ce sera donc un programme unique, bâti et enregistré une fois pour toutes, de même qu'un seul dictionnaire suffit à assurer la traduction d'une langue dans une autre de n'importe quel texte.

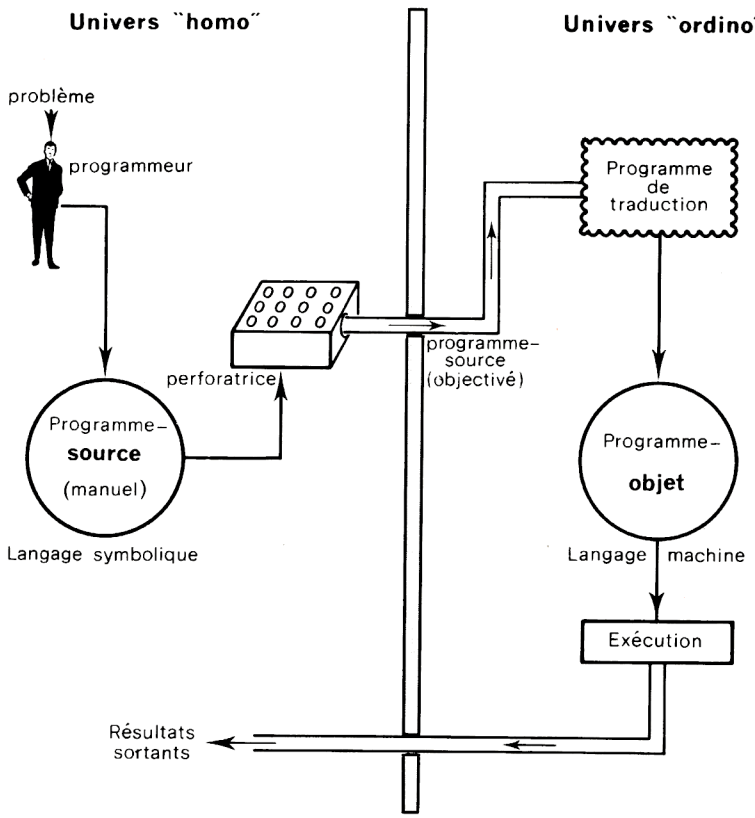
Cette idée va s'éclaircir à mesure que nous la fouillerons. Commençons par un peu de vocabulaire, ce qui est le début de toute science. Suivez, s'il vous plaît, sur la figure ci-contre.

On appelle « **programme-source** » le programme écrit par le concepteur avec des symboles spéciaux qu'il peut apprendre et retenir facilement parce qu'ils restent voisins de ses modes d'expression habituels. Ces symboles seront toujours les mêmes pour une machine donnée, mais ils peuvent bien sûr s'associer d'infinies manières différentes pour former les programmes-sources attachés aux différents problèmes.

Par exemple, si l'ordre « transférer de la mémoire vers l'accumulateur » possède, dans le code d'ordres de la machine avec laquelle on travaille, la représentation binaire 1001101001110, qui est donc son écriture en langage machine, je l'écrirai moi, simplement « TRA », utilisant les trois premières lettres du premier mot pour former une abréviation. Et si je veux spécifier de quelle case de mémoire il faudra faire partir le transfert, j'écrirai son adresse carrément en décimal, système de numération auquel je suis habitué, et non en binaire comme la machine le demande pour travailler. Ainsi, les programmes-sources écrits dans cette « *symbolique homo* » ressemblent exactement à ces programmes embryonnaires que nous avions barbouillés dans les chapitres précédents, lorsque justement nous ne nous préoccupions pas des tops-tips qui vont finalement exprimer nos instructions dans le langage machine.

Lorsque le concepteur a terminé l'écriture de son programme-source en langage symbolique, il faut d'abord le faire entrer en machine. C'est une gentille et patiente demoiselle qui s'en charge. Elle tape le texte-source sur un clavier spécial qui le transforme en une suite de perforations sur des cartes, ou de marques magnétiques sur une bande, après quoi la machine avale ces signes qu'elle sait lire (alors qu'elle ne sait pas lire l'écriture manuelle des hommes). Attention! A cette phase on n'a encore rien « traduit », bien qu'on ait engendré des tops et tips. Leur suite ne représente que le texte même du programme-source, sans aucun changement de sa structure. Par exemple, si dans le programme source figurait l'ordre « TRA », la dactylo aura

engendré le code binaire de T, puis celui de R, puis celui de A, mais pas du tout le nombre 1001101001110 qui représente l'ordre « TRA » dans le code de la machine. On peut dire qu'à la sortie de cette frappe, on a simplement le même programme-



source, mais « *objectivé* » dans des signes que la machine est capable de manier : c'est ce qui est représenté sur la figure par le tuyau supérieur qui achemine le programme-source *objectivé* vers le dedans de l'ordinateur.

Là on a fait aussi entrer le *programme de traduction et d'interprétation* qui, lui, aura été écrit directement en langage machine. Ce second programme va faire dérouler une série d'opérations qui auront pour effet de lire et comprendre le programme-source, et de réémettre ce que le concepteur veut que la machine fasse sous la forme de ce qu'on appelle le « **programme-objet** ». Celui-ci ne récite plus du tout en tops-tips le texte du programme-source tel qu'il fut écrit, mais aligne directement les ordres selon le code de la machine, et les données selon les adresses de la machine. La succession des ordres n'est plus la même, car les enchaînements et les priorités ne se présentent pas dans la machine de la même façon que dans le cerveau du programmeur. Aussitôt le programme-objet engendré par l'application du programme de traduction au programme-source, ces deux derniers sont expulsés de la machine où seul subsiste

le premier, qui est en état de tourner. En général, le programme de traduction regagne la bibliothèque où il repose, et le programme-source (objectivé) est recopié sur une bande magnétique pour être conservé. Le programme-objet aussi, d'ailleurs, car si on en a besoin une autre fois il sera directement accessible et on n'aura plus besoin de faire le détour par un programme-source et sa traduction.

Remarquez que la perte de temps qu'occasionne ce détour est largement compensée par le fait que le programmeur, n'ayant plus à connaître que des symboles d'un genre familier, se sent infiniment plus à l'aise pour concevoir et pour vérifier. Vous retrouvez l'idée de base que c'est l'homme qui constitue le maillon le plus cher de la chaîne, et que c'est donc lui qu'il convient de soulager.

Premier type de langage symbolique : le langage dit d' « assembleur »

Dans une première époque, cette programmation symbolique n'ambitionne encore qu'un confort intellectuel relativement limité. Les programmeurs acceptent de respecter la structure de la machine avec laquelle ils travaillent, c'est-à-dire surtout son code d'ordres. Simplement, au lieu de désigner ces ordres par leurs codes binaires, ils se permettent d'écrire, en alphabet *homo*, des lettres abrégeant le nom commun de ces ordres par une sorte de sténographie. Outre l'avantage de cette sténographie, les programmeurs se donnent aussi la latitude d'écrire les adresses et les données en décimal. Ils désignent aussi par des abréviations mnémotechniques les titres des sous-programmes existant en bibliothèque, par exemple « *cos* » pour un sous-programme permettant de calculer la fonction *cosinus* d'une variable. La sténographie est universelle pour éviter les particularismes linguistiques, mais, en vérité, elle dérive de l'anglais, langue mère de l'informatique.

▶▶ — Est-ce là tout ?

▶ — Vous pensez bien que non. Pour que le programmeur puisse écrire aisément ses instructions et qu'ensuite le programme de traduction ne rencontre aucune cause d'ambiguïté, il faut encore mettre en œuvre une série de « canons » : des étiquettes pour matriculer les pas du programme et parfois désigner leur nature; des conventions sur le rôle et le placement des parenthèses; des signes convenus pour marquer les débuts et les fins ainsi que les opérations; etc. Je ne vous développerai pas tout cet attirail de conventions, car elles appartiennent à l'art du spécialiste. Je m'attarderai toutefois un peu sur l'une d'elles, qui est intéressante philosophiquement parce qu'elle concerne un tour fondamental de la pensée.

Cette convention d'écriture s'appelle en termes savants « l'affectation d'une valeur à un identificateur ». Voyons ce que ce latin là veut dire.

Il arrive souvent qu'au cours d'un programme se retrouve, en plusieurs endroits, une quantité qui représente toujours la même chose mais avec des valeurs différentes. Ce fut le cas, au chapitre précédent, de la quantité que nous avions désignée par Z_n et dont les valeurs successives au cours des itérations N° $\ldots, n-1, n, n+1, \ldots$, s'approchaient de plus en plus de la racine carrée que nous cherchions. Pensez aussi à l'âge dans un calcul de probabilité de décès, qui signifie toujours la même chose pour tous les assurés mais prend pour chacun une valeur différente. En fin de compte, ces apparitions d'une entité unique à divers moments sous des valeurs diverses

caractérisent justement ce qu'on appelle une « ***variable*** ». Si vous saisissez bien cette notion de ***variable***, vous apercevez sans peine que pour la manier pratiquement il faut exécuter deux opérations intellectuelles distinctes. Dans la première on doit *définir* l'entité dont on parle et la désigner par un *symbole* (en général une lettre); et, dans la seconde, lui *assigner une valeur*.

Voici un exemple de la première opération :

« J'appellerai $\underbrace{(A)}_{\text{le symbole}}$ $\overbrace{\text{l'âge des assurés.}}^{\text{l'entité}}$ »

C'est le symbole, ici A, qui s'appelle l' « identificateur » (sous-entendu : de l'entité). Le reste de la définition s'appelle une « *déclaration* ». Plus exactement, on appelle ainsi, dans le jargon de la programmation, l'ensemble des précisions qu'on fournit sur le rôle que jouera l'identificateur, sur la manière dont on le traitera, le genre de grandeur que c'est (nombre, réel ou imaginaire, vecteur, matrice), etc. C'est de cela en effet que l'ordinateur a besoin.

Nous avons vu qu'il suffit souvent, *pour instituer une variable*, de lui réserver une case déterminée de la mémoire : toute valeur sera réputée être une valeur de cette variable dès l'instant qu'elle a été trouvée ou prise dans cette case là (ou dans un groupe de cases si la variable possède plusieurs dimensions). Cette définition de la variable *par son adresse* suffit en général à l'ordinateur, qui peut ignorer ce qu'elle représente dans l'univers sensible. Cela, le programmeur se l'inscrit, pour son compte, dans une marge spéciale de son rouleau de papier consacrée à ce qu'on appelle les « commentaires ». Ceux-ci servent seulement à soutenir sa pensée et n'entrent pas dans la machine.

Vient maintenant le second acte, le plus délicat et de haute teneur logique : il faut écrire d'une façon standard, d'une façon que le programme de traduction pourra comprendre, qu'on *assigne* telle ou telle valeur à une variable, donc à son identificateur. Vous pourriez penser que le vieux signe $= y$ suffit. Si par exemple je veux signifier que j'assigne la valeur 12 au descripteur Z, je pourrais écrire simplement :

$$Z = 12.$$

Cette écriture dissimulerait la vraie nature de l'acte que j'accomplis. Elle pourrait en effet signifier que j'ai fait des calculs sur la variable Z et que le résultat est 12. Alors que c'est *par ma volonté* que je fais $Z = 12$, par exemple quand je *dis* à la machine : exécute-moi tel calcul avec la valeur 12 de Z. En outre, que se passerait-il si la valeur que je veux assigner à une variable désignée par une lettre est elle-même désignée par une lettre? L'égalité $a = b$, au sens ordinaire, est *symétrique* entre les deux quantités qu'elle met en regard, tandis que je tiens à marquer une *dissymétrie* entre l'identificateur et sa valeur.

Dans certains langages, on conserve le signe $=$ mais on rompt sa symétrie par la simple convention que ce que l'ordinateur trouve *à gauche* du signe $=$ devra toujours être entendu comme l'*identificateur* d'une variable, tandis que ce qu'il trouve *à droite* sera toujours sa *valeur*. D'autres préfèrent détruire la symétrie plus carrément, par exemple en écrivant le mot sténographique EQU (abrégé de l'anglais « equal ») à la place du signe $=$, ou encore en lui rajoutant un graphisme, en général

deux petits points aux bouts des deux petites barres qui pointent vers l'identificateur, comme ceci

$$Z \coloneqq 12$$

Cette notation a l'avantage d'abord de laisser libre le signe = dans son emploi traditionnel, ensuite de permettre de donner de l'identificateur une définition plus complexe. On peut en effet placer à droite, non plus seulement une valeur, mais une *expression* complète, pouvant contenir à son tour d'autres identificateurs préalablement définis. Ainsi par exemple :

$$Y \coloneqq \frac{1}{2 + X}$$

Cela veut dire : on prendra pour valeurs de la variable Y celles que donne l'expression $\frac{1}{2 + X}$ lorsqu'on a assigné des valeurs à la variable X. Vous reconnaissez la notion de fonction, avec cette précision logique que l'expression $\frac{1}{2 + X}$ *institue* la variable Y. On peut même, au moyen de la notation $\coloneqq$, instituer des variables en écrivant à droite une définition formulée avec des *mots*, par exemple :

$Y \coloneqq$ racine la plus grande de telle équation

ou

$Y \coloneqq$ étiquette de la dernière instruction de branchement

ou

$Y \coloneqq$ le plus grand des nombres (U, V, W).

Une forme spécialement futée de l'emploi du signe = pour instituer des variables se rencontre dans les itérations, par exemple quand on lit

$$X = X + 1.$$

Ce serait évidemment absurde s'il fallait prendre le signe = dans le sens habituel. Mais si vous l'interprétez comme une définition de l'identificateur qui est à gauche par la valeur qui est à droite, cette écriture n'offre plus rien d'étrange, elle signifie simplement que la variable X prendra désormais la valeur obtenue en ajoutant 1 à la valeur qu'elle avait jusqu'alors. Autrement dit, que c'est $X + 1$ qui jouera dans la suite le rôle de X. Ce tour d'esprit, très fréquent dans le monde *ordino*, vous deviendra clair si vous imaginez que la lettre écrite *à gauche* caractérise une *adresse* de mémoire. L'écriture $X = X + 1$ signifie alors qu'on place dans l'adresse réservée à la variable X la *valeur* $X + 1$ écrite *à droite* et où, parce qu'on est à droite, la même lettre X représente, non plus une adresse, mais une valeur antérieurement prise par cette même variable.

Je ne vous en dirai pas plus sur les langages symboliques de cette veine. Leur traduction en langage machine n'est pas commode, mais au moins pas trop compliquée intellectuellement, parce que, en fin de compte, les symboles familiers qui sont utilisés par le programmeur correspondent un à un aux ordres et à la structure des mots que la machine sait traiter. On ajoute cependant au « programme de traduction » une tâche supplémentaire importante, qui est de *préparer la mémoire*. Il ne suffit pas en effet, pour que le programme-objet puisse tourner sans encombre, que le

programme-source ait été correctement traduit en tops et tips. Il faut encore qu'il spécifie où, dans la mémoire, seront logés les ordres, les données et les variables, et quelles zones on utilisera pour le transit des résultats intermédiaires. Cette partie du travail s'appelle « **l'assemblage** », et c'est pourquoi les langages symboliques du genre que je viens de vous décrire s'appellent des « *langages d'assembleur* ». Le programme de traduction capable de préparer aussi le mode de garnissage de la mémoire s'appelle un « **assembleur** ».

La conception d'un assembleur est fort difficile, car on exige qu'il sache meubler la mémoire d'une façon optimale *quel que soit le problème traité*. Il faut donc que l'auteur d'un tel programme d'assemblage identifie la liste des « situations » pouvant apparaître dans tous les problèmes, articule le programme assembleur de façon qu'il puisse reconnaître dans laquelle de ces situations se place le problème traité, et prévoie les consignes pour que le programme-objet assemble la mémoire en conséquence. En général, on est obligé de faire élaborer d'abord un programme-objet *préparatoire* pour « reconnaître le terrain » à traiter, dans une première passe du programme-source, après quoi une seconde passe effectue la traduction et fournit le programme-objet définitif avec assemblage optimum.

Vous sentez bien qu'on se situe là au plus haut niveau de l'art de la programmation. Aussi la création de l'assembleur unique qui permettra la traduction et l'assemblage de tous les programmes-sources, pour leur traitement dans un ordinateur donné, est-elle laissée aux soins du fabricant de l'ordinateur, qui vend ce programme en même temps que sa machine — (et que le reste du software de service). Les auteurs de ce programme de traduction et d'assemblage travaillent d'ailleurs en étroite symbiose avec les ingénieurs de la machine même, car celle-ci sera d'autant plus performante que l'écriture des programmes-sources sera plus aisée et leur traduction-assemblage plus rapide, plus sûre et plus astucieuse; et cette préoccupation contribue beaucoup à choisir tels ou tels dispositifs pour le traitement de l'information dans le sein de la machine. C'est là le point en lequel *le software détermine une partie du hardware*.

Les langages symboliques évolués

Les langages d'assembleur sont très commodes et servent toujours beaucoup. Outre le programme assembleur qui engendre les programmes-objets d'une machine, le fabricant de cette machine vend un petit opuscule qui précise les conventions à respecter pour écrire les programmes-sources. Il suffit en général de quelques jours pour maîtriser ces langages d'assembleur, et comme leur articulation logique et graphique est très proche des modes de pensée et d'écriture courants, même un usager qui n'est pas un programmeur professionnel pourra écrire rapidement des petits programmes valables. Voilà qui est évidemment très précieux. Mais c'est encore insuffisant si on veut atteindre le but qu'on s'était fixé à la fin du chapitre précédent et qui avait motivé toute la recherche de « langages » spéciaux.

▶ — Vous souvenez-vous de ce but?

▶▶ — Je crois bien. J'y pensais par devers moi, pendant que vous me parliez des langages d'assembleur, car je vois bien qu'ils ne le réalisent pas. Ce but était

de pouvoir commercialiser des programmes et sous-programmes vraiment *universels*, c'est-à-dire adaptables *tels quels* à *tous* les ordinateurs. Or, si j'ai bien compris, un langage d'assembleur reste lié à un ordinateur déterminé. Faudra-t-il que les créateurs de ce software commercial le rédigent en autant de versions qu'il existe de langages d'assembleur, donc d'ordinateurs, sur le marché?

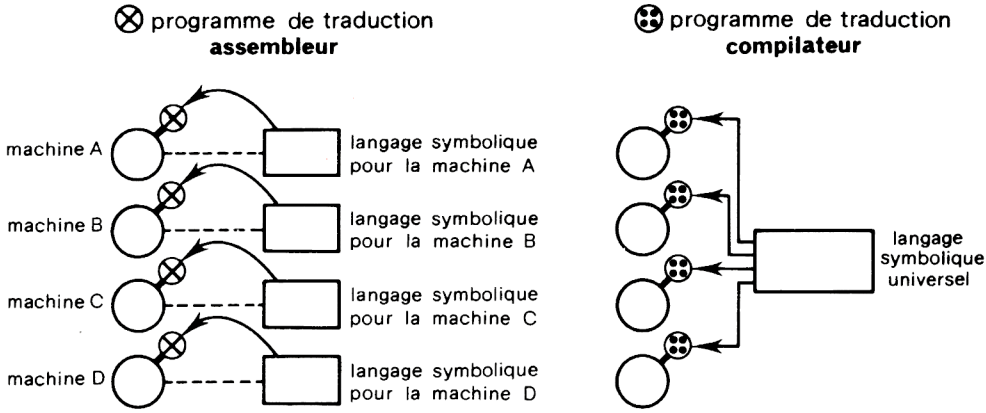
▸ — Tel est en effet le défaut qui subsiste dans ces langages symboliques dont la sténographie dépend de la structure de la machine qui les emploiera. Pour pouvoir y remédier, les informaticiens sont allés encore plus loin dans l'affranchissement des langages de programmation, c'est-à-dire qu'ils ont forgé des langages complètement indépendants des machines, qui expriment directement ce que le programmeur veut faire en épargnant à celui-ci le souci de savoir si les ordres correspondants figurent ou non dans l'attirail de l'ordinateur qu'on va utiliser.

▸▸ — Mais que se passe-t-il alors avec un ordinateur qui ne possède effectivement pas les ordres que le programmeur a écrits?

▸ — Réfléchissez que *tout* ordinateur mis sur le marché possède une gamme d'ordres suffisante pour accomplir *tous* les traitements imaginables. Plus ou moins astucieusement, bien sûr, mais complètement. Je vous ai déjà dit par exemple qu'une machine qui ne serait pas dotée d'un ordre spécial de multiplication pourrait néanmoins multiplier en accumulant des additions. Si donc j'écrivais, dans un programme que je destinerais à cette machine, l'expression $a \times b$, je pourrais encore me faire obéir à condition qu'un programme de traduction, lisant mon expression et comprenant qu'elle prescrit une multiplication, en déduise qu'il faut demander à la machine une série d'additions. C'est cela qui va me permettre d'écrire mon programme en ne pensant plus qu'à mon problème et ignorant complètement la machine qui sera appelée à le traiter. Celle-ci sera dotée d'un programme de traduction à ce point intelligent qu'il reformulera de fond en comble mes souhaits de façon qu'ils puissent être exécutés avec les ordres et les tailles de mots que la machine est capable de manipuler. En plus, bien sûr, ce programme prendra soin de meubler convenablement la mémoire, comme dans le cas de l'assembleur. Un tel programme, qui traduit un programme-source écrit dans un langage symbolique tout à fait universel en programme-objet pour un ordinateur donné, s'appelle un « **compilateur** ». La figure ci-contre illustre la différence entre les langages d'assembleur qui existent à raison d'un pour chaque machine, et un langage symbolique de hiérarchie plus élevée, qui est vraiment universel et indépendant de toute machine. Cependant, les programmes de traduction du genre « compilateurs », qui admettent à leur entrée des programmes-sources écrits dans un tel langage, restent encore liés à chaque machine par leur sortie, puisqu'on doit retrouver là un programme-objet écrit dans le langage de la machine. Chaque compilateur doit en somme comprendre le langage évolué de tout le monde et servir le dialecte primitif de son cuisinier propre.

Un langage de programmation symbolique de ce niveau s'appelle « évolué » ou « orienté vers les problèmes », ou encore « algorithmique » puisqu'il est ajusté de façon qu'on puisse écrire facilement les algorithmes sans s'occuper d'avance de la manière dont tel ou tel ordinateur les exécutera. Vous pensez bien qu'un langage algorithmique universel a besoin d'encore plus de conventions d'écriture précises que les langages d'assembleur. Les déclarations définissant les identificateurs, les étiquettes de pas de programmes, les titres de sous-programmes, les parenthèses,

affectations de valeurs à des variables, marques de début et fin, notations pour branchements, etc., doivent être employées selon des règles très soigneusement agencées, qui finissent par prendre l'allure d'une véritable syntaxe comme celle qui régit les



combinaisons des grandes catégories grammaticales (verbes, noms, articles, prépositions, etc.) dans les langues *homo*. En revanche, on se permet d'écrire les mots opératoires en clair, sans abréviations. Pour vous faire voir un peu de quoi a l'air un programme-source écrit dans un langage de ce type, je vous transcris ci-dessous deux petits morceaux, l'un en français, l'autre en anglais :

- Pour $N \leftarrow N + 1$ tant que $1/N \uparrow 2 \geqslant \varepsilon$ faire $S \leftarrow 1/(N + 1)$.
- If $(A > B)$ and $(C \leqslant D)$, then $X \leftarrow 0$, else $X \leftarrow 1$.

La première phrase signifie que, après avoir remplacé N par $N + 1$ et à condition que $1/N^2$ reste supérieur à ε (le symbole $N \uparrow 2$ veut dire N au carré), il faudra assigner à S la valeur $1/N + 1$. La deuxième phrase veut dire que si à la fois, au pas de programme où on est arrivé, A est supérieur à B et C inférieur ou égal à D, alors on assignera à X la valeur 0, dans le cas contraire on assignera à X la valeur 1. Les lettres N, S, A, B, C, D, X, qui interviennent là sont des identificateurs qu'il aura fallu définir au préalable par des déclarations en bonne et due forme.

Les échantillons que je viens de vous donner sont encore assez transparents. Mais pour écrire de façon ramassée des suites d'ordres plus complexes, on a été amené à charger ces langages algorithmiques de syntaxes assez lourdes, qu'on ne domine bien qu'après plusieurs mois d'apprentissage et un an de pratique. C'est le prix qu'on paye pour l'avantage de pouvoir exprimer tous les programmes imaginables dans un système unique de notations. Et ce n'est pas seulement l'étude de ce système qui est difficile, c'est aussi la confection des compilateurs correspondants. Ceux-ci doivent en effet commencer par faire une « *analyse syntaxique* » du programme-source pour reconnaître et délimiter les unités logiques du texte et leurs liaisons (quelle partie est une déclaration, de quoi est-elle suivie, etc.). Après quoi, le compilateur peut s'enquérir du *sens* des fragments ainsi séparés — c'est la phase d' « *analyse sémantique* ». Et ce n'est qu'alors que le programme-source peut se transposer en

programme-objet, avec à la clé l'inévitable meublement optimum de la mémoire. Un programme compilateur de haute qualité peut contenir jusqu'à 10 000 instructions!

Mais il y a plus grave que cette lourdeur. Le développement de ces langages supérieurement évolués s'est accompli de façon très anarchique, d'où résulte qu'il en existe aujourd'hui non pas, comme ce serait souhaitable, un seul vraiment universel, mais quatre grands : FORTRAN (FORmula TRANslator), COBOL (COmmon Business Oriented Language), ALGOL (ALGOrithmic Language), PL 1 (Programming Language n° 1), et vingt petits : BASIC, APT, GAP, LISP, IPL, SLIP, etc. Chacun de ces langages revendiquant pour lui les plus hautes vertus et les possédant en effet... dans une sphère spéciale. Dès lors, le rêve du langage universel dans lequel on peut écrire et vendre tous les programmes imaginables s'évanouit à nouveau! Pour alimenter le marché, il a fallu écrire les programmes préfabriqués dans les quatre grands langages algorithmiques actuellement en vogue, et la plupart des ordinateurs sont vendus avec des compilateurs permettant de traduire ces quatre langages vers eux. Le langage PL 1, créé par IBM et porté par la puissance exceptionnelle de cette firme, semble en passe de détrôner les autres. Mais une école de pensée concurrente fait tous les jours des progrès dans une autre voie.

Pour cette école, le langage algorithmique vraiment universel est une utopie. Elle préconise que chaque groupe d'usagers conçoive le langage le mieux adapté à ses besoins propres, un langage aussi simple que possible, mais très efficace pour un type de problèmes, puis *enseigne* à la machine avec laquelle ce groupe travaille à comprendre les programmes-sources écrits dans ces langages *ad hoc* d'atelier, c'est-à-dire que l'atelier en question forge aussi un programme de traduction. Si cet atelier veut communiquer un programme conçu dans son sein à des collègues extérieurs, il leur envoie la description du langage dans un opuscule. S'il veut bénéficier des grosses unités de software préparées par les grands centres d'informatique il leur envoie aussi cet opuscule et paye la recopie des dites unités dans son langage *ad hoc*.

Vous voyez que le marché mondial du software d'informatique ne risque pas d'être plus rationnel que celui de l'édition traditionnelle, où la littérature apparaît dans une trentaine de langues et une centaine de formats...

L'ordinateur renouvelle la grammaire

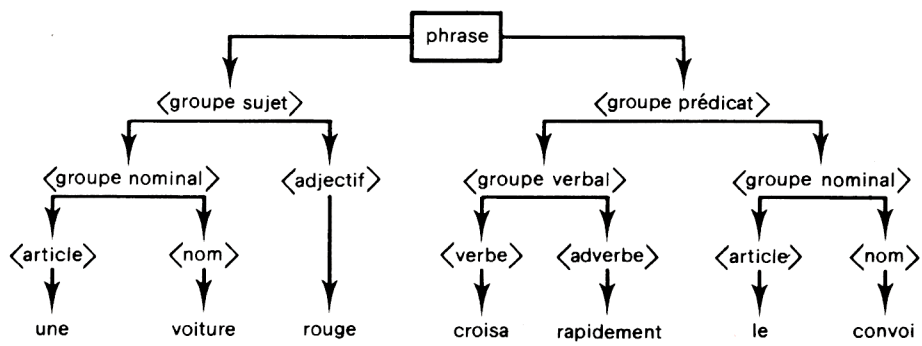
L'élaboration des langages de programmation de plus en plus subtils a eu une incidence en retour assez remarquable sur la linguistique. On s'est aperçu en effet que certaines des articulations qu'il fallait établir pour la programmation entre les grands concepts de déclarations, identificateurs, valeurs, séparateurs, sous-programmes, branchements, etc., étaient du même type que les articulations syntaxiques qui existent, dans les langues naturelles, entre les catégories de mots telles que verbes, noms, adjectifs, articles, etc. Or, avec les langages de programmation algorithmiques, on a vu qu'il fallait mettre au point, comme première étape du compilateur, une « *analyse syntaxique* » qui permette, en explorant les phrases écrites dans le programme-source, de reconnaître et d'isoler les catégories dont elle est

composée. On s'est avisé que cette méthode d'analyse permettrait peut-être d'éclairer aussi l'anatomie intime des phrases naturelles. C'est un peu comme des architectes qui, en bâtissant des maisons, retrouveraient certaines des normes que la nature a suivies pour bâtir des montagnes, ou comme l'urbaniste qui retrouve des lois de sociologie générale dans le microcosme qui vient peupler ses grands ensembles.

C'est ainsi qu'on est parvenu à réaliser cette structuration d'une phrase sous la forme d'un « arbre à embranchements logiques » dont rêvait de Saussure. Considérez par exemple la phrase :

« *Une voiture rouge croisa rapidement le convoi.* »

Si on la décompose selon le genre de syntaxe qu'on emploie dans les langages de programmation, on obtient le schéma arborescent suivant :



Dans ce schéma on a rangé les fragments de structure en descendant du plus général vers les plus spécialisés, et attendant les tout derniers morceaux, ceux qui ne se décomposent plus en sous-structures, pour les remplir avec les mots spécifiques qui composaient la phrase donnée. Les parenthèses anguleuses qui entourent les fragments de structures sont là pour qu'on distingue les *catégories* dont on parle des *mots* qui les désignent. On rencontre en effet en linguistique (comme dans bien d'autres domaines) la difficulté qu'il n'existe qu'une seule langue — le français que j'écris et que vous lisez — pour deux fonctions différentes : au premier niveau la fonction consistant à s'exprimer, au second niveau la fonction consistant à décrire la manière dont le premier s'articule. Autrement dit, on ne peut expliquer la structure du français... qu'en français! Alors qu'en bonne logique le mode opératoire d'une langue devrait être énoncé dans une langue différente, supérieure, directement intelligible, qu'on appelle « *méta-langue* ». Eh bien, les parenthèses pointues ont été inventées pour inviter le lecteur à se placer, en esprit, dans le cadre de cette méta-langue qui n'existe pas. Ainsi le mot « adjectif », dans son état brut, constitue simplement l'un des vocables de la langue française, qu'un ordinateur interprétera en lisant *a*, *d*, *j*, *e*, *c*, *t*, *i* et *f*. Alors que si vous voyez écrit <adjectif> avec les parenthèses pointues autour, vous devez imaginer que l'auteur a voulu vous inviter à penser à la catégorie syntaxique qui devrait porter un nom à elle dans une éventuelle méta-langue, mais qu'en l'absence de celle-ci on vous désigne avec le vocabulaire de la

langue existante; et un ordinateur se contentera d'ouvrir la catégorie, où il attendra un adjectif particulier.

Une flèche, dans le schéma arborescent, indique que la catégorie d'où elle part peut être *remplacée* par — ou *décomposée en* — la catégorie, ou la suite de catégories, où elle aboutit. Les catégories *terminales*, c'est-à-dire celles qu'on trouve *au bout* de ces générations successives, sont celles qu'on remplace par des mots spécifiques appartenant à ces catégories; les mots étant les atomes finaux, insécables, de la langue. Et la tâche de ce que j'ai appelé plus haut « l'*analyse syntaxique* » d'une phrase consiste à remonter des mots dont elle est composée aux structures qui les embrassent, c'est-à-dire à retrouver le schéma arborescent précédent *de bas en haut*.

Les actes générateurs qui conduisent d'une catégorie à une ou plusieurs autres doivent naturellement se conformer à des *règles* qui autorisent certains remplacements et en interdisent d'autres, et dont l'ensemble constitue la « grammaire » de la langue. On est très loin encore d'avoir pu faire la théorie de grammaires aussi complexes que celles des langues naturelles. A défaut, les théoriciens se sont beaucoup amusés avec des langues artificielles. C'est à qui inventerait une grammaire plus curieuse que les préeédentes. De toutes ces langues a émergé en position dominante une classe particulièrement simple, dont les grammaires s'appellent « *non contextuelles* » (en anglais, « *context free* »). Ces grammaires se distinguent par le fait que les règles permettant le passage d'une catégorie génératrice en « amont » vers une ou plusieurs catégories engendrées en « aval », sont exclusivement attachées à la première et ne dépendent pas de tout ce qui peut la précéder dans une phrase, c'est-à-dire de son « contexte ». Il n'en est jamais ainsi dans les langues naturelles, constructions tortueuses où les mots se connectent de manières différentes selon leurs antécédents (par exemple, en mode affirmatif, un verbe peut être suivi directement de son complément — « elle couvre ses jambes » — tandis qu'en mode négatif, c'est-à-dire s'il est précédé de « ne », le verbe doit conduire d'abord à la préposition « pas » — « elle ne couvre pas ses jambes »). Cependant *les grammaires « context free » s'avèrent être justement celles des langages de programmation évolués*, et c'est pourquoi elles ont tellement intéressé les théoriciens. Ceux-ci ont démontré une foule de jolies propriétés que possèdent les langages à grammaire non contextuelle, en particulier le théorème fondamental que *leurs phrases admettent toujours une analyse syntaxique non ambiguë*. C'est ce théorème qui explique qu'on puisse bâtir, pour traduire les programmes-sources écrits dans un langage algorithmique, ces programmes que nous avons appelés des « compilateurs » et qui doivent commencer par une analyse syntaxique du programme-source.

Les coups de sonde que les langages de programmation ont permis de donner dans la mystérieuse texture des langues naturelles sont très intéressants, surtout parce qu'ils ont renouvelé les méthodes d'étude d'une discipline qui s'encroûtait; mais ils n'ont permis encore que de jeter des lumières très modestes sur cette plus étonnante création de l'homme. Songez que déjà la syntaxe n'est qu'un chapitre mineur, puisqu'elle ne concerne que la *forme* des constructions et non leur *signification*, qui ressortit à la « *sémantique* » et est bien plus importante (par exemple, la syntaxe seule n'aurait rien à redire à une phrase telle que « la pluie enjambe plus de soupirs que les graines du dictionnaire », car elle est parfaitement construite, mais n'a aucun sens). En outre, même dans son cadre, la syntaxe ne peut rendre compte de phénomènes linguistiques tels que l'identité profonde entre la phrase citée plus haut (une

voiture rouge croisa rapidement le convoi) et la suivante : le convoi fut croisé; c'était par une voiture rouge rapide. En résumé, il reste infiniment plus difficile de comprendre comment j'arrive à me faire comprendre de vous, que de comprendre comment un programmeur se fait comprendre par son ordinateur...

On a revu d'un œil critique le programme écrit dans le *langage machine* que l'ordinateur comprend, c'est-à-dire comme suite de mots binaires, et qui s'appelle le *« programme-objet »*. Où vous vous êtes vite convaincu qu'il n'y a rien de plus assommant que d'écrire le traitement qu'on veut faire exécuter par l'ordinateur directement sous la forme d'un programme-objet. Et où l'envie vous est venue d'écrire votre programme dans un *langage symbolique* plus proche de vos habitudes, sous une forme qu'on appelle le *« programme-source »*. Où vous avez compris ensuite que cette paresse est possible à condition qu'on ait préparé une fois pour toutes un *« programme d'interprétation »*, qui permet à l'ordinateur lui-même de *traduire* le programme-source en un programme-objet. Où vous avez appris enfin que l'ensemble des symboles et des règles avec lesquels on peut écrire les programmes-sources s'appelle, non sans raison, un *« langage symbolique »*; qu'il y a des langages symboliques dits *« d'assembleur »*, bâtis autour de chaque machine, et des langages symboliques dits *« évolués »* ou *« universels »* indépendants des machines mais que chaque machine doit savoir traduire au moyen d'un programme d'interprétation propre qu'on appelle son *compilateur*. Après avoir jeté un rapide coup d'œil sur ces drôles de langages qui ont donné ses ailes à la programmation, vous avez appris qu'il y en a de nombreux de par le monde, et qu'à les bâtir on a gagné des vues nouvelles et pénétrantes sur nos langages naturels.

Deuxième partie

le hardware

Les grandes divisions du hardware

La commutation

La mémorisation

Puck

Les périphériques

Un peu d'histoire et un peu de prospective

Les grandes divisions du hardware

La « quincaillerie » des ordinateurs se subdivise assez clairement en trois grands domaines :

- *Les circuits servant à la* ***commutation*** *appelés aussi* « ***circuits logiques*** ».
- *Les éléments servant à la* ***mémorisation*** *appelés aussi* « ***mémoires*** ».
- *Les organes* ***périphériques*** *servant aux entrées et sorties.*

Les circuits de **commutation** sont ceux qui assurent le traitement *dynamique* des tops et tips *fugitifs*. Ces circuits engendrent ou éliminent les impulsions, les aiguillent vers telle ou telle voie, les décalent dans le temps ou l'espace, les comptent, assemblent ou séparent. Ce sont eux aussi qui exécutent les opérations arithmétiques proprement dites. Ils sont essentiellement faits de *composants électroniques*, et nous verrons (enfin!) comment une certaine combinaison de ces composants permet de « comprendre » les messages qui lui sont présentés. Le mot de « commutation » est employé pour désigner toutes les fonctions qu'assurent ces circuits, parce qu'elles reposent toutes en fin de compte sur cet acte simple qu'on appelle « commuter » et qui consiste à envoyer une impulsion d'un côté ou de l'autre. Les commutations dans les circuits électroniques peuvent se dérouler à la fantastique vitesse de 10 millions par seconde, alors que les commutations vieux jeu, par relais électromécaniques, plafonnent à quelques dizaines par seconde.

Les éléments de **mémorisation** sont ceux qui accueillent les tops et tips quand on les leur envoie et les conservent intacts aussi longtemps qu'on ne les leur redemande pas. Nous verrons que ce sont essentiellement des éléments *magnétiques*, mais ils ont besoin de circuits électroniques de commutation pour acheminer les tops et tips vers une adresse déterminée ou les extraire d'une adresse déterminée. Nous n'appelons ici « **mémoire** » que l'organe capable d'absorber et délivrer l'information *à la même ultra grande vitesse que celle des commutations*, organe fait de composants parfaitement statiques où rien ne bouge mécaniquement.

Les organes **périphériques** constituent les indispensables chaînons entre l'homme et l'ordinateur. Ils sont, en règle générale, électromécaniques et permettent à l'homme d'insérer l'information dans la machine sous la forme qui convient à la machine, et à la machine de sortir l'information vers l'homme sous la forme qui convient à l'homme. Comme ces échanges s'effectuent grâce à des mouvements de supports matériels, ils sont des milliers et des milliers de fois plus lents que les commutations intérieures. Les périphériques sont donc essentiellement des rouages d'adaptation

qui connectent deux univers de biologies très différentes, un peu comme nos sens qui connectent le monde extérieur avec notre cerveau. On les appelle de plus en plus des « **interfaces** », nom un peu barbare mais qui dit bien ce qu'il veut dire et va sans doute entrer rapidement dans le langage courant, car il a des applications bien au-delà de l'informatique; par exemple, le Parlement est (ou devrait être) une « interface » entre le peuple et le gouvernement... Il se trouve d'autre part que certains organes périphériques peuvent servir de mémoires auxiliaires, beaucoup plus *lentes* certes que la mémoire statique, mais ayant une *capacité* considérablement supérieure. Dans ce rôle de mémoires auxiliaires, ces périphériques deviennent donc des parties intégrantes du traitement.

Avant de vous décrire ces trois domaines du hardware, il faut encore que je vous explique le mot de « ***signal*** » qui va surgir constamment sous ma plume. Vous savez peut-être qu'en électricité on distingue :

— le ***courant*** électrique qui ***circule dans*** les conducteurs (fils métalliques) et se mesure en « *ampères* »
et

— la ***tension*** électrique (également appelée « différence de potentiel ») qui ***existe entre*** deux bornes et se mesure en « *volts* ».

Le *courant* est un peu l'analogue de l'écoulement d'eau dans une canalisation ou dans une chute d'eau, alors que la *tension* ressemble à la différence de pression, ou de niveau, entre deux points de l'écoulement. Lorsqu'on étudie les circuits en détail, il est essentiel de distinguer entre ces deux formes de l'électricité. Mais lorsqu'on veut seulement préciser ce qui arrive à l'information, il suffit de savoir si elle est présente ou absente en un certain point du système, sans qu'on eût à examiner si elle se manifeste sous la forme d'un courant ou celle d'une tension. Pour englober les deux cas, on emploie le vocable *unique* de « ***signal*** ». Ce mot est bien choisi dans une science de l'information, puisqu'il appartient au vocabulaire de la communication : l'éclat du phare est un signal, un clignement d'yeux chez les amoureux, la bougie déplacée derrière une fenêtre chez les espions, la présence d'un coquillage fossile dans un terrain chez les géologues, etc., sont des « signaux ». En outre, le mot couvre aussi bien l'absence que la présence d'une excitation possible ou d'une manifestation attendue. Nous appellerons donc dans la suite « ***signal*** » aussi bien un top qu'un tip, sous la forme indifféremment d'une impulsion de courant ou d'une impulsion de tension.

Après avoir passé en revue les pièces de ces deux grands jeux de dominos électroniques que sont la commutation et la mémorisation, nous les assemblerons pour former le squelette d'un petit ordinateur imaginaire, tout simple mais contenant les traits essentiels des grandes machines, et qu'à cause de son charme de lutin agile et serviable, nous appellerons « Puck ». Après quoi nous rendrons visite aux périphériques. Dans un dernier chapitre de cette seconde partie, nous jetterons un coup d'œil sur le passé et l'avenir du hardware d'ordinateur.

... Et si vous n'aimez pas vous pencher sur le fonctionnement interne des gadgets, vous pouvez, à la rigueur, sauter toute cette deuxième partie, *en admettant* que les actions dont on aura à parler dans la troisième partie peuvent effectivement s'accomplir dans le sein de l'ordinateur.

4

La commutation

Le moyen de commuter

La clé fondamentale de tout ce que vous allez apprendre à propos du hardware s'appelle l' « électron ». Un courant électrique n'est qu'un flot d'électrons dans un conducteur. Vous savez sans doute que l'électron est le grain ultime de l'électricité, qui porte une charge *négative* égale environ à $-1{,}5 \cdot 10^{-16}$ coulomb et dont la masse est environ $6 \cdot 10^{-31}$ kg. D'après le premier chiffre, il faut qu'il s'écoule environ 10^{16} électrons (soit dix millions de milliards de grains) en une seconde pour que soit passé un courant de 1 ampère. Mais le second chiffre est beaucoup plus important. Une masse de 10^{-31} kg est si fantastiquement petite que cette petitesse échappe à l'esprit. Elle a pour conséquence capitale que l'électron *n'oppose qu'une inertie infime*, *infime*, *infime* aux sollicitations qu'on lui fait de se déplacer, et c'est cela qui nous donne le moyen d'établir, éteindre, aiguiller des signaux avec des vitesses prodigieuses et des dépenses minimes d'énergie.

Mais les flots d'électrons qui forment les courants dans les fils métalliques se prêtent mal à subir des actions « *contrôlantes* ». Vous rencontrez ici l'un des maîtres mots de la technique — aussi bien la technique du maniement de la matière que la technique du maniement des hommes. On ne peut prétendre connaître la vie et les choses tant qu'on ne comprend pas ce mot dans toute sa plénitude. La notion complète de « contrôle » comprend les aspects suivants :

— *la chose contrôlée* est une grandeur, un phénomène, un mouvement, une passion, qui se déroule d'une certaine manière;

— *la chose contrôlante* est un signal que le *contrôleur* peut actionner à sa guise pour influencer dans le sens qu'il souhaite le déroulement de la chose contrôlée;

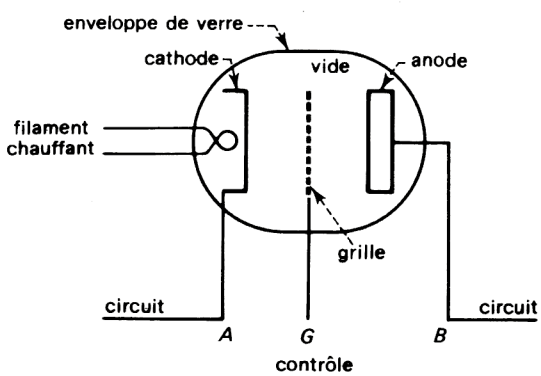
— pour qu'il y ait vraiment « contrôle », il faut en général que l'incidence de la chose contrôlante sur la chose contrôlée soit rapide par rapport aux rythmes de déroulement de cette dernière, qu'elle puisse être modelée à discrétion, et enfin — condition importante mais trop souvent passée sous silence — qu'elle n'exige qu'un déploiement d'énergie faible comparé à celui que déploie la chose contrôlée.

Un prototype de contrôle est l'aiguillage d'un train. Par un geste très simple et qui lui coûte à peine 10 joules d'énergie, l'aiguilleur envoie très vite et où il veut (dans la limite des rails installés), cinq mille tonnes de ferraille roulant à 100 km/heure

c'est-à-dire déployant une énergie de 50 millions de joules. On ne dirait pas qu'on « contrôle » la marche des trains si, pour en dévier un, on devait lui atteler une locomotive de même puissance et attendre une heure qu'elle ait traîné ses wagons sur un autre itinéraire.

Eh bien, les flots d'électrons dans les conducteurs ne sont pas *contrôlables* facilement dans le sens précédent. Le seul moyen qu'on ait d'influencer le courant qui passe dans un tronçon conducteur, c'est de changer la tension aux bornes du tronçon, et alors le courant ne sait changer que d'une façon naïvement proportionnelle, et on doit dépenser une énergie égale à la fraction qui se modifie. C'est pourquoi les techniciens avaient compris dès le début qu'un vrai contrôle ne pouvait s'obtenir qu'en faisant parcourir aux électrons un tronçon *non conducteur* et en insérant sur le parcours une sorte de *vanne* qui pourrait commander le flux instantanément par un réglage ne consommant pas d'énergie. L'idée est donc, comme dans les écoulements liquides, d'avoir *trois* bornes : deux pour l'entrée et la sortie, et une pour la commande. Un tel tronçon peut être très petit et s'insérer dans un parcours métallique plus long; il suffit alors à contrôler le courant dans la totalité du parcours.

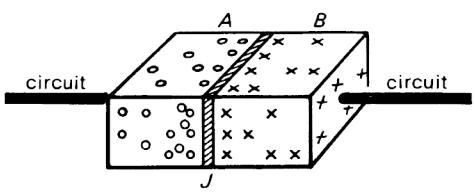
Au début de ce siècle, Lee de Forest imagina le premier type de tronçon capable de remplir ce programme. La clé de l'affaire était de *sortir des conducteurs*, où les électrons se pressent toujours en foule dans tous les points, pour faire travailler des électrons *spécialement engendrés*, et cela dans un parcours *vide* (car sinon les électrons se perdent très vite par leurs chocs avec les molécules d'air). Lee de Forest prenait donc une enveloppe de verre bien vidée et scellée, et y engendrait un flot d'électrons en chauffant de l'extérieur un morceau de métal appelé « *cathode* ». Ce flot se dirigeait vers un autre morceau de métal qu'on appelait « *anode* » (ou « plaque ») et dans le parcours on interposait ce qu'on appelait une « *grille* », qui, comme ce nom l'indique, était une troisième électrode de métal, percée de trous pour laisser passer les électrons. On démontre facilement qu'en modifiant la tension appliquée à la grille, on peut moduler instantanément et à discrétion le flot des électrons allant de la cathode à l'anode, sans dépenser plus de la millionième partie de l'énergie que peut véhiculer le flot d'électrons, car la grille elle-même n'est frappée que par quelques rares électrons s'égarant sur les bords des trous, et ne véhicule donc qu'un courant négligeable. L'ensemble s'appelle une « lampe électronique », ou un « tube à vide », ou un « tube » tout court quand il n'y a pas d'ambiguïté. Conceptuellement,

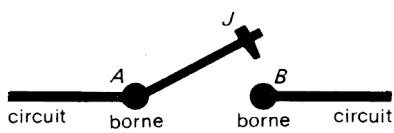


ce dispositif réalise le rêve du « contrôle » d'une façon qui n'a jamais été dépassée quant à la perfection des effets cherchés. C'est lui qui a rendu possible le développement prodigieux de l'électronique. Mais il souffrait de défauts qui allaient se révéler graves lorsqu'il fallut l'appliquer à l'ordinateur. D'abord il faut chauffer la cathode au rouge, ce qui consomme toujours une fraction de watt, et quand on met plusieurs milliers de lampes ensemble on crée un véritable four. Ensuite le verre, le métal scellé, le vide, forment une construction fragile qui, sauf précautions très coûteuses, ne fonctionne jamais plus de 1 000 heures sans panne, et dont toujours une ou plusieurs sur tout lot de 1 000 sont défectueuses. Il en résultait que, dans cette forêt de milliers de lampes qui constitue un ordinateur, les malheurs étaient très fréquents. Avec les premiers calculateurs à lampes, on était content lorsque le temps moyen de bon fonctionnement entre pannes atteignait 10 heures!

La scène changea du tout au tout lorsqu'après la guerre on inventa un autre type de tronçon contrôlable, basé sur les propriétés presque miraculeuses de certains cristaux appelés « ***semiconducteurs*** ». Il y avait longtemps que les chercheurs caressaient l'idée de faire travailler les électrons dans le sein d'un cristal au lieu du vide. Un cristal en effet contient des atomes disposés selon un réseau très dense et très régulier, qui offrent en principe aux évolutions des électrons une ambiance favorable. Mais, en pratique, pour que ce réseau ne dérange pas les évolutions souhaitées, sa régularité géométrique et sa pureté chimique doivent être telles qu'on désespérait d'en jamais fabriquer d'assez parfaits : les erreurs sur le placement des atomes ne doivent pas dépasser en effet un cent millième de leurs distances, et il ne doit pas y avoir plus d'un atome étranger sur dix mille millions!

Or, vers la fin des années 40, on inventa subitement des procédés que je ne peux pas vous décrire ici (*) et qui permirent de préparer des cristaux (de germanium et de silicium) respectant ces fabuleuses performances de qualité. On apprit du même coup à introduire dans le réseau cristallin, à bon escient, des ***impuretés*** *exactememt dosées*, qu'on appelle des « *dopants* ». Par exemple : un atome de dopant par mille millions d'atomes du cristal. Et dès lors apparut vite le fait fondamental suivant : si dans un morceau de « cristal parfait », on établit deux compartiments *A* et *B* en dopant chacun d'eux avec deux impuretés différentes astucieusement choisies, il se forme à leur jonction *J* une sorte de « *membrane à électrons* » aux propriétés étonnantes qui vont en faire un organe de contrôle remarquable. Si on applique au compartiment *B* une tension *d'un certain signe* par rapport au compartiment *A*, la membrane se laisse traverser par un flot abondant d'électrons qui croît exponentiellement



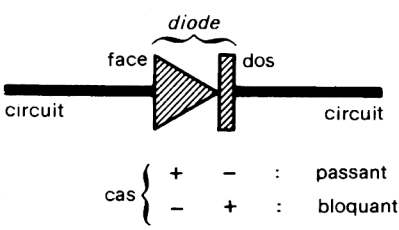


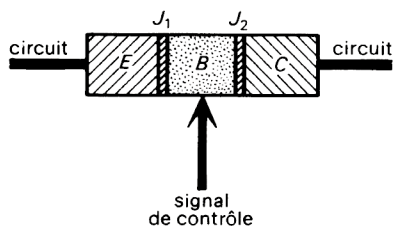
(*) Vous trouverez des détails dans un autre volume de cette collection, « Comprendre l'électronique ».

avec la tension; tandis que, si celle-ci vaut zéro ou n'importe quoi de l'*autre* signe, la membrane devient hermétique et aucun courant ne peut plus aller de *A* en *B*. Tout se passe comme si on avait un contact *J* actionné par le *signe* de la différence de potentiel entre *A* et *B* : si ce signe est « direct », le contact se ferme, et il circule autant de courant qu'on veut; s'il est « inverse », ou que la différence de potentiel soit nulle, le contact s'ouvre et aucun courant ne peut plus passer. On a donc un contrôle par « *tout ou rien* », la chose contrôlante n'étant capable que de deux états — les deux signes positif et négatif — et la chose contrôlée de même — courant ou pas courant. Ce dispositif s'appelle une « ***diode*** » parce qu'il n'y a que *deux* bornes (le préfixe « ***di*** » signifiant deux et le suffixe « ***ode*** » commun à « électrode », « anode », « cathode », etc., voulant dire « borne »). C'est l'équivalent électronique d'un contact, mais alors qu'un contact mécanique ne peut s'ouvrir et se fermer qu'au plus quelques dizaines de fois par seconde, une diode électronique peut suivre sans rechigner jusqu'à des dizaines de millions de changements de signe par seconde.

On représente une diode conventionnellement par le schéma ci-dessous, où la pointe du triangle faisant flèche indique le sens « passant », c'est-à-dire celui qui permet de véhiculer du courant lorsque le côté amont de la flèche est positif par rapport à son côté aval. Cette pointe peut être considérée comme la « face » de la diode tandis que le côté opposé est son « dos ». Une diode est donc *passante* si le potentiel de sa face est plus grand que celui de son dos, et *bloquante* dans le cas contraire.

La diode ne réalise pas tout à fait l'idéal du tronçon de contrôle puisque la borne sur laquelle s'applique la grandeur contrôlante n'est pas physiquement séparée des deux bornes entre lesquelles passe la grandeur contrôlée. Elle n'est d'ailleurs pas tout à fait un outil pour contrôler la *valeur* d'un signal, c'est plutôt un outil pour contrôler le *cheminement* d'un signal, une barrière levée ou fermée, bref un organe d'*aiguillage*. Mais on passe de là facilement à un vrai tronçon capable de contrôler une *valeur* en rajoutant un troisième compartiment, dopé lui aussi d'une manière idoine (en général avec le même dopant qu'à l'autre bout, mais sous une concentration différente). On crée ainsi une seconde jonction J_2 et un dispositif vraiment « ***triode*** ». Sur la figure, les trois compartiments sont désignés par les lettres *E*, *B*, *C*, parce qu'ils s'appellent « *émetteur* », « *base* » et « *collecteur* ». En jouant sur les valeurs et les signes des trois signaux appliqués aux trois compartiments, on peut obtenir une multitude d'effets intéressants. En particulier, lorsque les compartiments extrêmes *E* et *C* sont convenablement polarisés, on peut faire en sorte que du courant entre eux passe ou ne passe pas, déployant une certaine énergie dans le circuit, en appliquant au compartiment *B*, avec une énergie pratiquement nulle,



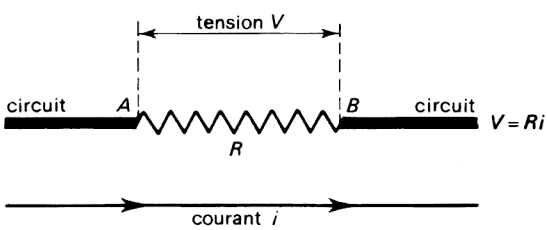


une tension d'un signe ou de l'autre. On obtient ainsi un organe de contrôle au sens pur du terme où, au lieu d'opérer comme avec la diode dans le flot même, on travaille sur lui à partir d'une position latérale, et en outre avec une dépense d'énergie très inférieure à celle que déploie le flot contrôlé. On peut énoncer cette vertu en disant qu'on obtient un effet combiné, non seulement d'aiguillage comme dans la diode, mais encore d'*amplification*, puisque l'impulsion circulant dans le circuit où est inséré ce tronçon peut être considérablement plus puissante que l'impulsion qui la contrôle.

Vous êtes, cher lecteur, en présence du fameux « ***transistor*** ». Ce nom englobe aussi bien les diodes que les triodes, et bien d'autres structures inventées dans les dernières années en multipliant les jonctions et disposant des compartiments de façons variées. La flore des transistors possède aujourd'hui une centaine de types. Ils se basent tous sur les procédés miracles qui permettent de fabriquer des cristaux quasiment parfaits et de les doper avec une finesse et une précision fabuleuses. Ils sont infiniment plus petits que les tubes à vide — le rapport des volumes étant de 1 à 10 000 — et en même temps beaucoup plus robustes et souples. Ce sont des minuscules grains de cristal travaillant à l'*air libre* — adieu le vide! — et *à froid* — adieu le four! Et ceux qui ont passé victorieusement les essais de vérification sont d'une fiabilité sensationnelle, leurs caractéristiques ne bougeant plus pendant des dizaines d'années. En outre, l'industrie moderne — dont je vous ai déjà dit qu'elle a réussi là son plus extraordinaire exploit — produit désormais ces petits grains à des prix de plusieurs dizaines de fois inférieurs à ceux des tubes à vide. Ce féerique transistor qui permit la réduction par cent de la taille et l'amélioration par cent de la qualité de tant d'équipements électroniques (dont les petits récepteurs radio qui portent son nom) constitue la brique de base de toute la circuiterie de commutation d'un ordinateur moderne, qui en contient souvent des centaines de mille. C'est grâce à lui que l'ordinateur a pu devenir une machine sûre au point de vue technique, maniable au point de vue encombrement et accessible au point de vue économique.

Les circuits logiques élémentaires

Vous allez voir tout de suite combien de jolies choses ont peut faire avec de simples diodes. Je vous rappelle d'abord ce qu'est une « *résistance* ». C'est un tronçon de *conducteur* métallique (pas de cristal!) qui n'a pas d'autres propriétés basiques que tous les conducteurs, sauf qu'on le construit volontairement de façon que le courant rencontre une gêne à le traverser, de sorte qu'il perd de son potentiel comme



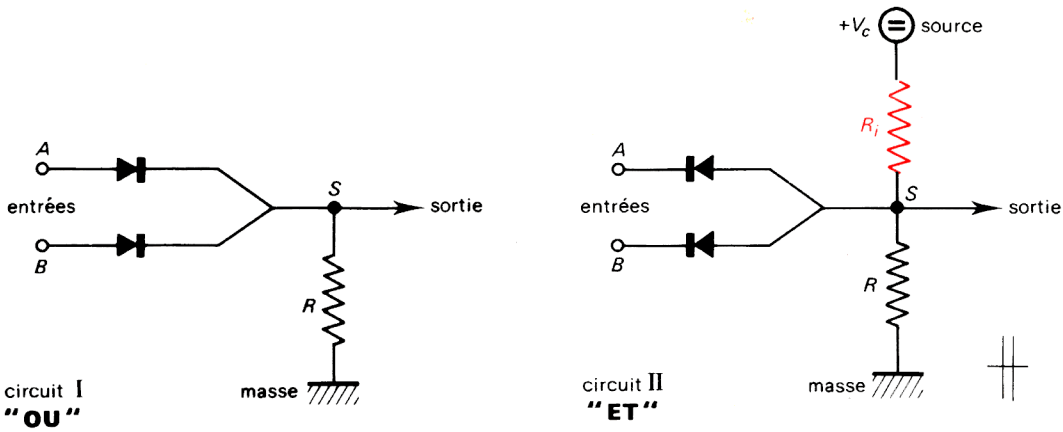
un fluide perd sa pression quand il s'écoule dans un tuyau frottant. La tension V qui apparaît ainsi aux bornes de la résistance de valeur R est simplement proportionnelle au courant i qui la traverse, avec R comme coefficient de proportionnalité :

$$V = Ri.$$

Tout morceau de conducteur métallique possède une certaine résistance, mais tellement petite qu'on peut et doit en général négliger la chute que souffre le potentiel lorsque du courant le traverse. Il faut préparer un morceau de conducteur d'une façon spéciale pour que cette chute soit notable. En bref, on place une résistance dans un parcours lorsqu'on veut y faire « débiter » du courant de façon à récolter une tension appréciable entre ses bornes. Si une résistance insérée entre deux points A et B d'un circuit ne débite pas, c'est-à-dire n'est pas parcourue par un courant, la tension qui règne en A se transpose sans chute en B. On représente une résistance par un petit serpentement de zigzags.

Ceci dit, considérons d'abord le circuit I dessiné page 93. La « résistance de charge R » a son point bas à la masse. Son point haut S est connecté vers la droite à une *sortie* menant à des circuits d'utilisation, et vers la gauche à deux *entrées* A et B à travers deux diodes qui présentent leurs *faces* aux entrées et leurs *dos* à S. Tant qu'il n'y a pas de signal ni sur l'entrée A, ni sur l'entrée B, les diodes ont leurs faces et leurs dos au potentiel nul, elles sont donc « bloquées », rien ne se passe, le point S et la résistance de charge R restent morts, et il n'y a pas de signal à la sortie. Si au contraire une seule entrée, que ce soit A ou que ce soit B, est excitée par un signal constitué par une tension positive, la diode qui tourne sa face vers cette entrée devient *passante*, du courant y circule de l'entrée vers S, s'écoule de S vers la masse à travers la résistance de charge R, le potentiel de S, point haut de R, s'élève puisqu'il doit s'établir une chute de potentiel entre ces deux bornes de R, et ce potentiel en S qui n'est plus nul devient du signal présent à la sortie. Il en va de même, *a fortiori*, si les deux entrées sont excitées à la fois. On dimensionne naturellement les éléments de façon que, lorsque du courant passe dans la résistance R, le potentiel V' de son point haut S ne s'élève pas *au-dessus* de la valeur positive V que possèdent les signaux à l'entrée car sinon la différence $V - V'$ entre les potentiels sur les faces et les dos des diodes deviendrait négative et les diodes se bloqueraient à nouveau. En fait, le circuit travaille avec un V' inférieur mais presque égal à V, le résidu $V - V'$ restant juste assez positif pour que les diodes restent passantes. Vous voyez qu'elles servent en quelque sorte de *chemins d'injection* vers la résistance de charge. Ces chemins sont fermés, laissant la sortie à zéro, lorsqu'il n'y a aucun signal à l'entrée et il suffit, pour que du signal apparaisse à la sortie, qu'un seul chemin d'injection s'ouvre, donc qu'une seule entrée soit excitée par du signal. On dit que c'est un circuit « OU » parce qu'il fonctionne selon la condition suivante :

il y a du signal à la sortie	*il n'y a pas de signal à la sortie*
si	**si**
il y a du signal { (ou) sur l'entrée A, (ou) sur l'entrée B, (ou) sur les deux entrées.	il n'y a pas de signal ni en A ni en B.



Considérons maintenant le circuit II de la figure ci-dessus où vous voyez encore une « résistance de charge » R placée entre la sortie S et la masse, et deux chemins liant S, vers la gauche, aux deux entrées A et B à travers deux diodes qui présentent cette fois leurs *dos* aux entrées et leurs *faces* à S. En plus, une source de tension positive V_c peut alimenter le point S à travers une autre résistance R_i ce qui tend à faire circuler du courant issu de la source vers S. Les diodes ne sont plus des chemins d'*injection* vers S, mais au contraire des chemins de *drainage* de ce courant hors de S. Et le courant est en effet drainé vers la gauche tant que l'une au moins de ces diodes s'en laisse traverser, ce qui a lieu tant que l'une au moins n'a pas de signal à l'entrée. Car alors ce courant aura en effet provoqué, en traversant la résistance R_i, une chute de tension qui ramène le potentiel en S à une faible valeur, juste assez pour laisser passante celle des deux diodes qui ne supporte pas de signal entrant et qui assure le drainage (ou les deux diodes s'il n'y a de signal entrant sur aucune d'elles). Ce reflux du courant vers la gauche interdit la formation d'un potentiel appréciable en S, donc d'un signal de sortie. Pour *rompre* cette condition, il faut donc *fermer* les *deux* chemins de drainage possibles, et cela se produit s'il y a sur les deux entrées *à la fois*, c'est-à-dire sur les *deux* dos des deux diodes, du signal positif rendant leurs faces négatives par rapport aux dos. Dans ce cas toute échappée vers la gauche est interdite au courant qui ne peut plus emprunter que l'enfilade des deux résistances R_i et R. Il est facile de dimensionner celles-ci de façon que le potentiel qui se forme alors en S soit tout juste un peu inférieur à l'amplitude V des signaux, ce qui suffit à laisser les diodes bloquées. On appelle ce circuit un circuit « ET » parce qu'il fonctionne selon la condition suivante :

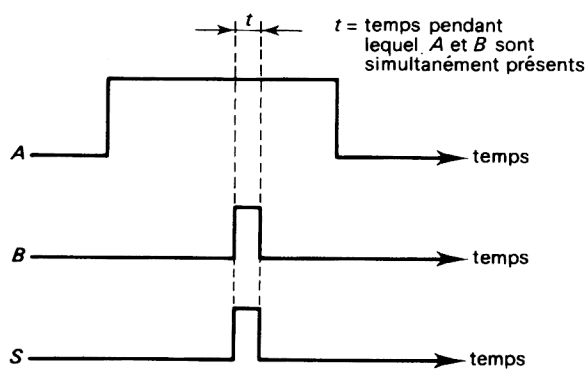
il y a du signal à la sortie	*il n'y a pas de signal à la sortie*
si	**si**
il y a du signal à la fois sur l'entrée A (et) sur l'entrée B.	• il n'y a de signal ni en A ni en B; • il y a du signal seulement en A; • il y a du signal seulement en B.

On peut résumer les fonctionnements de ces deux circuits comme suit :

— dans le circuit « OU » les diodes tendent à *injecter* du signal *vers la sortie S*; pour que le signal y paraisse en effet, on doit donc rendre cette injection *possible*, et pour cela → il suffit qu'une ***ou*** l'autre des entrées *A* et *B* ait du signal;

— dans le circuit « ET », les diodes tendent à *drainer* du signal *hors de la sortie S*; pour que le signal paraisse néanmoins en *S*, on doit donc rendre ce drainage *impossible*, et pour cela → il faut que l'entrée *A* ***et*** l'entrée *B* aient toutes deux du signal.

Il est à peine exagéré de dire que quand on a compris cette manière élégante de jouer avec les aptitudes d'aiguillage des diodes, on a compris l'essentiel du hardware de la commutation. On désigne les deux circuits I et II par les conjonctions « OU » et « ET » surtout lorsqu'on a en vue leurs fonctions « *logiques* ». D'un point de vue plus *technique*, on appelle souvent le circuit « OU » un circuit de « *mélange* », parce qu'il excite la sortie de quelque façon que se « mélangent » les entrées, seule l'absence de toute entrée entraînant l'absence de sortie. Tandis que le circuit « ET » s'appelle aussi un circuit de « *coïncidence* » parce qu'il faut que coïncident deux présences de signal sur les deux entrées pour qu'il y ait une sortie. On l'appelle encore une « *porte* » parce qu'on peut considérer que l'impulsion de sortie est celle qu'on applique à l'*une* des entrées et qu'on lui ouvre le passage lorsque l'*autre* entrée est également excitée. Observez à ce propos que si *B* est une impulsion courte et *A* une impulsion de longue durée embrassant l'époque où se produit *B*, il n'y aura de sortie que pendant le court intervalle d'existence de *B*, puisque ce n'est qu'alors que les deux signaux sont présents *à la fois*. Un circuit « ET » permet donc en quelque sorte de découper une tranche, à un instant donné, dans un signal de longue durée. Un modèle mécanique familier de cette fonction de coïncidence est l'ensemble objectif-obturateur d'un appareil photographique : l'objectif est frappé par la lumière permanente d'une scène (signal *A*) que l'obturateur admet dans l'appareil seulement pendant le temps où il est ouvert (signal *B*).



On peut naturellement, en multipliant les diodes, admettre *plusieurs* entrées et faire ainsi le mélange par « OU », ou la coïncidence par « ET », d'un certain nombre de signaux à la fois. On peut aussi aménager plusieurs sorties. Ces extensions finissent par poser quelques problèmes techniques, car les imperfections des diodes (dont je ne vous ai pas parlé mais qui bien entendu existent) s'accumulent. On y remédie par les mesures que je vous ai déjà mentionnées : on amplifie et redresse les signaux qui se sont dégradés au moyen de triodes supplémentaires. Les diodes elles-mêmes peuvent d'ailleurs être remplacées par des triodes. Je n'insiste pas, c'est du souci technique.

Incursion dans l'algèbre booléenne

Par contre, je ne résiste pas, ayant décrit les deux circuits « logiques » fondamentaux, au plaisir de vous promener un instant dans ce qu'on appelle l' « **algèbre de Boole** ». Non pas que vous en ayez vraiment besoin par la suite, mais parce qu'on en parle beaucoup, et que c'est bien joli. Revenons donc aux schémas de la page 93.

Décidons de désigner les signaux pouvant apparaître aux diverses bornes *A*, *B*, *S*, *par ces lettres A*, *B*, *S mêmes*. Ces lettres deviennent du coup des « *variables* » (ou plutôt les symboles de variables), mais restreintes au monde binaire de l'*ordino*, c'est-à-dire qu'elles ne peuvent prendre que les deux seules valeurs 0 et 1. Désignons aussi par les symboles :

$$\cup \qquad \text{et} \qquad \cap$$

les opérations « OU » et « ET » que les deux circuits exécutent sur les signaux qu'on leur applique. Cela permet d'exprimer par les relations suivantes les sorties qu'ils donnent en fonction des entrées qu'ils reçoivent :

$$\text{pour circuit I (circuit « OU ») : } S = A \cup B,$$
$$\text{pour circuit II (circuit « ET ») : } S = A \cap B.$$

Ceux d'entre vous qui ont parcouru la fameuse « théorie des ensembles » (*) auront reconnu dans $\cup$ et $\cap$ les deux opérations ensemblistes appelées « *union* » et « *intersection* », la première consistant à rassembler tous les éléments qui appartiennent ***ou*** à l'ensemble *A*, ***ou*** à l'ensemble *B* (ou aux deux), la seconde consistant à rassembler seulement tous les éléments qui appartiennent à la fois à l'ensemble *A* ***et*** à l'ensemble *B*. Dans notre cas, *A* et *B* étant des variables « booléennes » — ou binaires, c'est la même chose — susceptibles de prendre seulement les valeurs 0 et 1, ces deux opérations, effectuées par les deux circuits, se conforment aux règles des tableaux ci-après; on les appelle aussi « tableaux de vérité » parce qu'ils trouvent une application en logique pure, où *A* et *B* jouent le rôle de deux assertions telles que :

A : la jeune femme (dont on va parler) est jolie,

B : la jeune femme (dont on va parler) est coquette.

(*) Voir par exemple, dans cette collection, le volume « Comprendre les mathématiques ».

	Si $A =$	et $B =$	alors $S =$	Règles traduisant le tableau
tous les cas possibles pour A et B	0	0	0	$0 \cup 0 = 0$
	0	1	1	$0 \cup 1 = 1$
	1	0	1	$1 \cup 0 = 1$
	1	1	1	$1 \cup 1 = 1$

Tableau de l'opération "OU" ($\cup$)

	Si $A =$	et $B =$	alors $S =$	Règles traduisant le tableau
tous les cas possibles pour A et B	0	0	0	$0 \cap 0 = 0$
	0	1	0	$0 \cap 1 = 0$
	1	0	0	$1 \cap 0 = 0$
	1	1	1	$1 \cap 1 = 1$

Tableau de l'opération "ET" ($\cap$)

Si vous désignez par « 0 » le fait qu'une assertion est *fausse* (pour tel sujet particulier) et par « 1 » le fait qu'elle est *vraie*, vous constaterez facilement que les tableaux précédents indiquent quand sont vraies et fausses les assertions combinées « A ***ou*** B » ($A \cup B$) d'une part, « A ***et*** B » ($A \cap B$) d'autre part, en fonction de la vérité ou fausseté des assertions séparées. Ainsi, supposons que Marie-Chantal soit laide. Alors l'assertion A, pour elle, est fausse, et vaut donc 0. Supposons d'autre part que Marie-Chantal soit coquette : B est vraie, c'est-à-dire vaut 1. Il en résulte que l'assertion combinée « A ou B », c'est-à-dire Marie-Chantal est ou jolie *ou* coquette, est vraie, car $0 \cup 1 = 1$; tandis que l'assertion « A et B », c'est-à-dire « Marie-Chantal est jolie *et* coquette » est fausse car $0 \cap 1 = 0$.

Les combinaisons $\cup$ (« ou ») et $\cap$ (« et ») constituent les deux opérations fondamentales qu'on exécute sur deux variables booléennes. Une autre opération fondamentale, mais qu'on exécute sur *une seule* variable A, consiste à la transformer en son contraire : c'est-à-dire à engendrer une autre variable, qu'on désigne par la même lettre, mais surmontée d'une barre, donc $\overline{A}$, qui est égale à 1 si A vaut 0 et à 0 si A vaut 1. Cette opération s'appelle indifféremment « *inversion* », « *complémentation* » ou « *négation* » et s'effectue très simplement par un transistor triode, dans un circuit que je ne vous dessinerai pas.

A partir de ces trois opérations de base :

« ou » (ou encore « union ») qui à partir de A, B engendre $A \cup B$,
« et » (ou encore « intersection ») qui à partir de A, B engendre $A \cap B$,
« non » (ou encore « inversion ») qui à partir de A engendre $\overline{A}$,

on peut, en combinant le fruit d'une opération faite sur deux variables avec une troisième, et ainsi de suite, effectuer des opérations composites de plus en plus complexes sur un nombre de plus en plus grand de variables, dont l'expression forme des chaînes de plus en plus chargées. Ainsi l'opération composite

$$\overline{(A \cup B)} \cup (\overline{C} \cap \overline{D})$$

signifie qu'après avoir fait l'union de A, B, on inverse le résultat et qu'on le combine par « ou » au résultat que fournit l'intersection des inverses de C et D. Il faut naturellement, pour manipuler ces expressions et « calculer » correctement avec des variables,

respecter une série de règles que je ne vous détaillerai pas mais dont les tableaux donnés plus haut pour définir les opérations $\cup$ et $\cap$ sont des prototypes. Divers théorèmes permettent de transformer ces expressions, par développement ou par contraction, de façon à les ramener à des expressions plus commodes. L'ensemble de ces règles et de ces théorèmes forme ce qu'on appelle l' « *algèbre de Boole* », du nom du mathématicien anglais qui le premier rendit du lustre à la numération binaire et calcula systématiquement avec des variables assujetties à ne jamais prendre que les deux valeurs 0 et 1.

Boole avait introduit cette algèbre pour pouvoir plus facilement et sûrement combiner des propositions logiques. Lorsque les ordinateurs la remirent en honneur pour étudier le traitement de signaux représentés par des variables binaires, plusieurs auteurs découvrirent qu'au lieu de désigner les deux opérations de base par les graphismes $\cup$ et $\cap$, Boole les avait désignés par les *mêmes signes* que les opérations arithmétiques, à savoir $+$ pour l'union et $\cdot$ pour l'intersection. Ils firent donc de même, et ainsi apparurent dans le monde des traités où on pouvait lire ces formules apparemment insolites :

$$1 + 1 = 1,$$
$$0 \cdot 1 = 1.$$

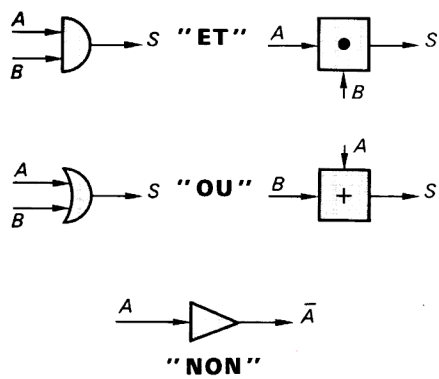
On cultiva cette impression d'étrangeté dans l'esprit du public en lui parlant d'additions mystérieuses qui ne faisaient plus la somme de 1 plus 1 égale à 2, et de multiplications révolutionnaires pour lesquelles le produit de 0 par quelque chose n'était plus égal à 0. Il n'y a là qu'un mauvais jeu sur des signes. C'est par pure convention qu'on reprend les signes $(+)$ et $(\cdot)$. En algèbre de Boole ils désignent non pas l'addition et la multiplication ordinaires de nombres ordinaires, mais des combinaisons toutes différentes de nombres restreints au champ binaire. Toute l'étrangeté se dissipe si on conserve, pour désigner ces combinaisons, les signes $\cup$ et $\cap$, car il n'y a rien de bouleversant à ce que des opérations autres que $+$ et $\cdot$ obéissent à d'autres lois. On pourrait très facilement dire que « le chariot traîne le cheval » si on appelait chariot ce que nous appelons cheval et cheval ce que nous appelons chariot. Cela ne signifierait pas que nous avons découvert des animaux de trait curieux qui ont avec des voitures étonnantes des rapports inverses de ceux qui nous sont familiers.

Toute expression booléenne portant sur des variables A, B, C, ..., peut être « incarnée » par les opérations *physiques* faites sur des signaux binaires au moyen des circuits élémentaires « et », « ou », « non ». Inversement, on a démontré que toute suite d'opérations physiques faites sur des signaux binaires, peut se ramener à une suite d'opérations d'unions, intersections et inversions, suite représentable par une expression booléenne. La totalité des traitements de signaux qui, dans les ordinateurs, tombent sous le titre « **commutation** » du présent chapitre, sont de ce type. Considérez par exemple la spécification suivante : « Je veux qu'un signal S apparaisse en tel endroit si en tel autre il y a un signal A, qu'en tel autre il n'y a pas un signal B, qu'en tel autre il y a un signal D en même temps qu'apparaîtrait un signal C sur l'union de A et B, mais ce signal D disparaissant par contre s'il y a un signal E présent en tel autre endroit, etc. ». On peut compliquer à loisir les conditions et conditions de conditions, il existera toujours une expression booléenne qui donnera S en fonction de A, B, C, D, ..., et par suite il existera toujours un circuit logique,

fait d'un assemblage plus ou moins touffu des circuits de base, qui « réalisera » cette spécification. Vous comprenez pourquoi on peut dire que Boole avait, sans s'en douter, forgé l'outil algébrique qui permet de « ***formaliser*** » tous les circuits de commutation des ordinateurs... 100 ans avant que ceux-ci apparaissent!

Les circuits logiques

Malheureusement, dans l'état actuel de la théorie, l'écriture booléenne d'une opération composite qu'on souhaite effectuer sur des signaux binaires ne permet pas de découvrir, par des réductions systématiques, quelle architecture de circuits logiques simples peut réaliser cette opération *de la façon la plus économique*. Cette tâche dite de « minimalisation » reste du ressort de l'ingéniosité du fabricant. Et celui-ci, pour bien « sentir » les combinaisons les meilleures de circuits, a besoin de les dessiner sur le papier. Il serait toutefois trop encombrant de dessiner les circuits électroniques eux-mêmes. Et ce serait inutile, vu que tout technicien saura les réaliser dès l'instant qu'on lui aura spécifié les opérations logiques qu'ils doivent accomplir. *Il suffit donc de représenter ces dernières*, et à cet effet, on a adopté des graphismes aujourd'hui standardisés dans le monde entier. Les voici :

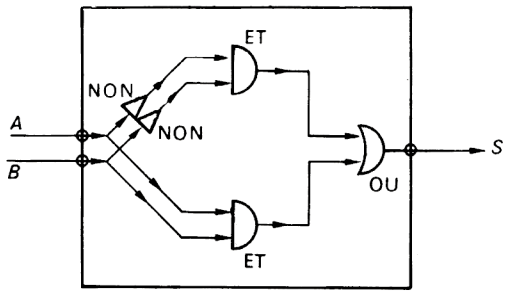


Les *schémas « ronds »* sont à préférer lorsqu'on peut dessiner les connections sur des lignes parallèles.

Les *schémas « carrés »* ne sont à employer *que* lorsque les connections se multiplient, car ils donnent alors la possibilité de faire déboucher les entrées n'importe où sur les trois côtés du carré qui ne vont pas vers la sortie.

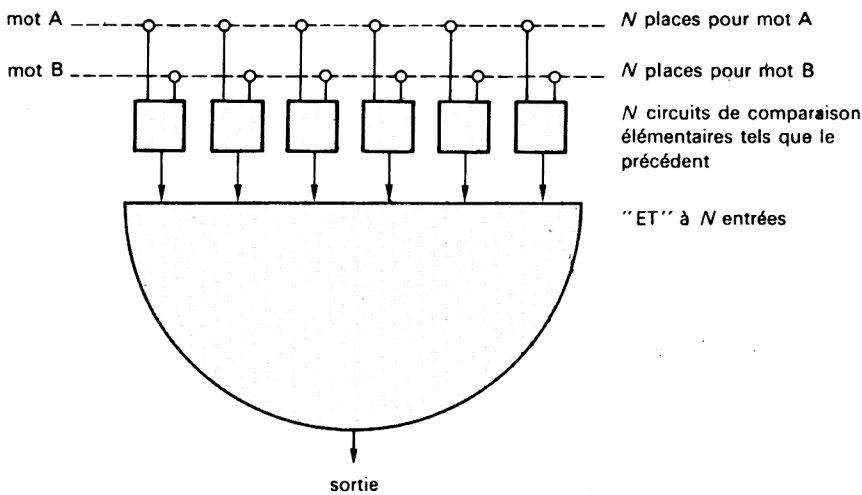
Un schéma d'ordinateur contient des milliers de ces petits carrés ou demi-lunes interconnectés. Je vais vous présenter quelques échantillons extrêmement simples qui vous illustreront des opérations composites importantes et vous donneront le plaisir de faire un petit bout de voyage à l'intérieur du hardware.

Je voudrais d'abord monter un circuit de *comparaison* qui me donne du signal si deux signaux *A* et *B* sont les mêmes (deux tops ou deux tips) et pas de signal s'ils sont différents. Je songe tout de suite à employer un « ET », car il fournira 0 si *A* et *B* sont différents et 1 si *A* et *B* sont tous deux des 1. Mais si *A* et *B* sont tous deux des 0, le « ET » unique fournira aussi 0, alors que je demande un 1. Qu'à cela ne tienne, je prends un second « ET » et je l'excite en dérivation par les mêmes signaux *A* et *B*, mais à travers deux inverseurs. Si *A* et *B* sont des 0, il y aura des 1 à la sortie des inverseurs, c'est-à-dire deux signaux sur les deux entrées du second « ET »,



donc du signal sur sa sortie. Tandis que le premier « ET », qui est frappé directement par des 0, reste passif. Et cela me suffit pour animer ma sortie *S* si je prends celle-ci à travers un « OU » qui admet sur son entrée les deux sorties des deux « ET ».

Si *A* et *B* sont différents, les deux « ET » reçoivent, chacun, des signaux non coïncidents, et ne livrent donc rien au « OU » final, de sorte qu'on n'a rien en *S*, comme on le souhaitait. Si on enveloppe le tout dans une boîte unique ayant ses entrées en *A* et *B* et sa sortie en *S*, on a fabriqué un *circuit de comparaison* selon mes vœux.



Une fois qu'on a résolu ce problème pour *deux* signaux binaires, il est facile de faire la comparaison *lexicographique* de deux mots entiers comportant *N* places. Il suffit que les *N* couples de digits occupant les places homologues des deux mots entrent dans *N* comparateurs simples qui débiteront chacun du signal à leur sortie si, et seulement si les digits en question sont identiques.

Si donc j'admets ces *N* sorties dans un grand « ET » à *N* entrées, j'aurai un signal à la sortie de ce dernier, qui sera la sortie générale, si, et seulement si les deux mots sont identiques digit à digit, c'est-à-dire « lexicographiquement ».

Je me tourne maintenant vers le problème suivant, qui est capital et que j'ai jusqu'ici passé sous silence : comment, au juste, accède-t-on à un casier de mémoire ?

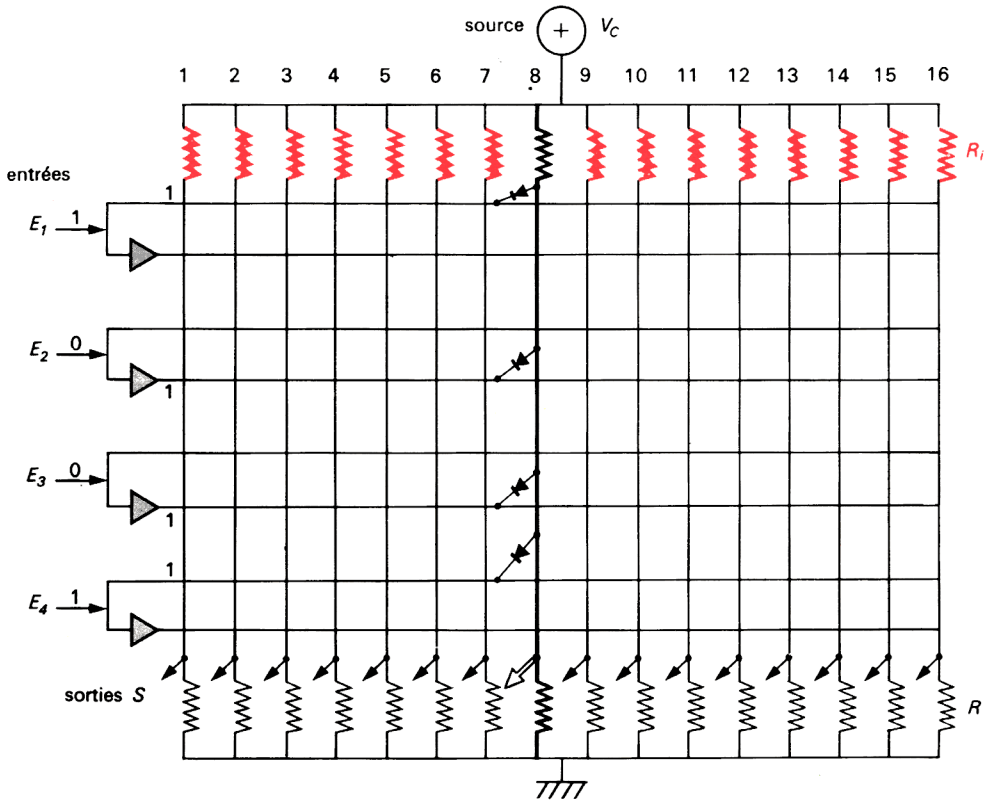
Le casier occupe un certain *lieu* où il faut aller porter des *impulsions*, et vous ne connaissez que l'*adresse* de ce lieu, c'est-à-dire un *nombre* (en fait deux nombres x et y) sous la forme d'un mot binaire. Il faut donc transformer en quelque sorte ce *nombre* en *positionnement*. C'est la tâche même de ce qu'on appelle un « *central d'aiguillage* » comme ceux qui contrôlent les réseaux téléphoniques. Dans ce cas aussi vous n'avez que l'adresse (téléphonique) de votre correspondant sous la forme d'un *mot* (qui actuellement comporte 7 digits décimaux). Par votre cadran, vous indiquez ce mot au central. D'autre part, votre correspondant est situé en un certain *endroit* de l'espace accessible par une *ligne* (physique) qui est l'analogue du *casier* (physique) de la mémoire d'un ordinateur. Et le central, recevant de vous une adresse, doit s'arranger pour aller convoyer les signaux sur celle des 10 000 lignes de son ressort qui le joint à l'endroit portant cette adresse.

Les centraux d'aiguillage du téléphone résolvent ce problème par des organes tournants qu'on appelle des « sélecteurs ». C'est beaucoup trop lent pour le monde *ordino* qui demande une solution électronique permettant de réaliser un aiguillage en une microseconde. Cette solution, très élégante, est fournie par un dispositif en matrice dont j'ai représenté ci-contre un exemple.

Supposons, par exemple, qu'à l'entrée on donne le mot binaire 1001, égal à *9*, et qu'il faille donc exciter le *neuvième* des fils de sortie.

La clé de la solution est l'observation simple mais illuminante que si on appelle E_1, E_2, E_3, E_4, ..., les digits successifs des mots, on peut toujours interpréter tout mot codé en tops et tips comme l'*intersection* de *tops seuls* (de 1) en traitant les *tips* (les 0) comme les inverses de tops. Ainsi 9 = 1001 peut s'écrire $E_1 \cap \overline{E_2} \cap \overline{E_3} \cap E_4$, où les valeurs 0 qu'ont les deux digits du milieu ont été considérées comme les inverses de valeurs 1. D'après cette manière de voir, je dois faire apparaître du signal sur la ligne N° 9 s'il y a à la fois à l'entrée du signal E_1, du signal $\overline{E_2}$, du signal $\overline{E_3}$ et du signal E_4. A partir de là, la construction du réseau est évidente. Au travers de 16 fils de sortie dessinés verticalement et au bout desquels je dois pouvoir recueillir les signaux (il y en a 16 pour pouvoir correspondre à tous les nombres possibles de 4 digits), je tends non pas *quatre* fils horizontaux pour les 4 digits, mais *8*, à raison de *2* fils par digit, celui du haut pour porter un top lorsque le digit est 1, et celui du bas pour porter un top lorsque le digit est 0, c'est-à-dire pour porter l'inverse du digit donné. Ces inverses sont aisément fabriqués par des circuits « NON » situés en amont. Pour exciter le fil de sortie N° 9 lorsqu'on applique le nombre 9 = 1001, je dois donc former, ***sur ce fil***, l'*intersection* des signaux arrivant par les fils E_1 haut, E_2 bas, E_3 bas, E_4 haut. Rien de plus simple : je réalise un « ET » multiple ayant le fil N° 9 pour sortie et les 4 fils précédents pour entrées. Il suffit à cet effet de placer selon le dispositif que vous connaissez, — et reconnaissez sur la figure — des *diodes drainantes* aux croisements correspondants, ainsi que la résistance de l'alimentation tout en haut (avec sa source de tension V_c) et la résistance de charge tout en bas. Et le tour est joué : si j'applique le mot 1001 sur les entrées E, du signal apparaît sur le fil N° 9.

L'aiguillage voulu est donc réalisé en brochant le réseau maillé des fils avec des diodes placées dans des sens convenables à des croisements idoines. Vous n'aurez aucune peine à trouver les croisements où il faut embrocher des diodes « drainantes » pour exciter, mettons le 5^e fil de sortie lorsqu'on applique le nombre 5, qui se repré-



sente par le mot 101. Et vous observerez que tous ces brochages ne se gênent nullement les uns les autres, c'est-à-dire qu'ils peuvent être exécutés et fonctionner en même temps côte à côte (ce dont on profite bien sûr pour réunir toutes les sources en une seule source et toutes les masses en une seule masse). Une fois garnie de ses diodes une matrice unique ayant, mettons 256 fils de sortie, aiguille du signal sur le fil adéquat N° n (si n est compris entre 0 et 255), dès qu'on affiche le nombre n sur 8 fils d'entrée (8 parce que $256 = 2^8$).

Nous avons donc complètement résolu notre problème. Et en fait nous avons résolu un problème encore plus général, car rien ne nous oblige à n'exciter, avec un nombre, que le fil dont le numéro est égal à ce nombre. Si je veux que l'application de tel *mot* excite tel *fil* que j'ai convenu d'avance, et même un groupe de fils, je peux très facilement l'obtenir en embrochant des diodes en conséquence dans mon réseau maillé. Je dispose donc d'un organe que je peux fabriquer à ma guise, de telle façon qu'il envoie des impulsions où je veux selon l'information que je lui applique. Bien entendu la correspondance entre cette information et la position du ou des fils que celle-ci excitera, si elle reste à ma discrétion *au moment où je fabrique la matrice*, ne sera plus modifiable *après*, elle se sera *incarnée* « *ne varietur* » *dans un certain plan de brochage*. Mais j'ai une entière liberté dans la conception de ce plan.

Quant au mot appliqué à la matrice de diodes, rien non plus ne nous restreint au cas où il est *numérique*. Il peut fort bien être *logique*, c'est-à-dire représenter un

ordre (ou toute autre information qui a été codée en tops-tips). Dès lors l'aiguillage planifié par un certain brochage de la matrice s'interprète comme suit : je me suis arrangé pour que, lorsque j'affiche tel ordre, des impulsions apparaissent sur tels fils. Or, ces impulsions à leur tour vont pouvoir « travailler » en allant ouvrir ou fermer, conditionnellement ou non, divers autres chemins, ou en allant se combiner de diverses façons dans des circuits logiques. On peut donc dire que la matrice d'aiguillage est devenue un *interpréteur* : grâce à son brochage sur mesure, un ordre va pouvoir provoquer toute suite d'effets voulus de moi en des lieux préétablis. C'est exactement ce que réalise le dressage d'un animal.

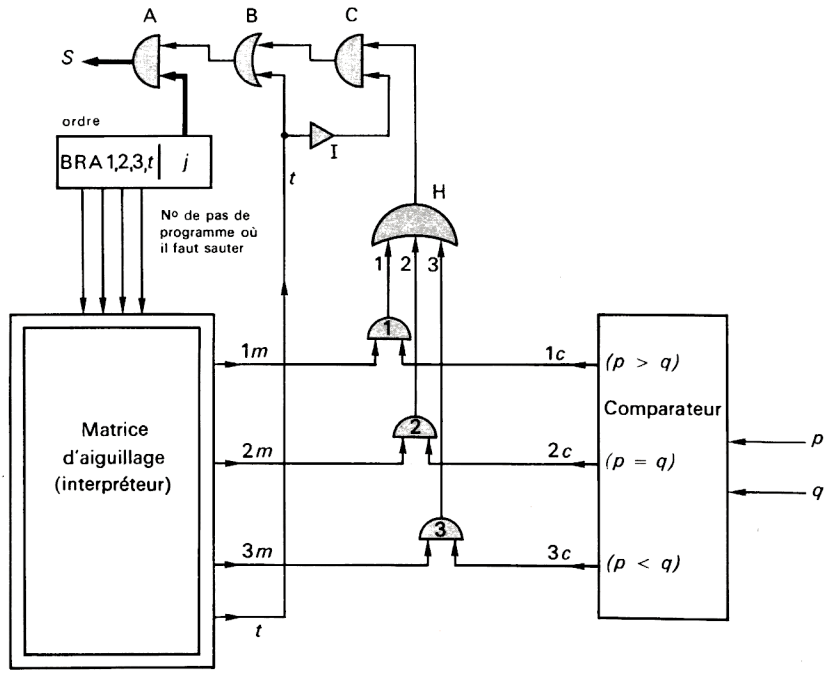
Nous sommes tout près d'éclaircir dans quel sens exact on peut dire qu'un ordinateur « comprend » un programme. La lumière sur ce point deviendra complète au moment où nous apprendrons à connaître Puck.

Le circuit du branchement conditionnel

Une application immédiate de la matrice d'aiguillage fonctionnant en « interpréteur d'ordres » est le circuit de saut conditionnel qui permet d'appeler un pas quelconque préétabli j du programme si un certain test, comparant par exemple deux nombres p et q, a fourni l'un des résultats : 1 ($p > q$); ou 2 ($p = q$); ou 3 ($p < q$). Supposons qu'on ait émis l'ordre « BRA 1 $\rightarrow j$ ». La partie de ce mot logique qui constitue l'ordre proprement dit sera en fait un mot en top-tip, mais sur la figure je l'ai simplement représentée par « BRA 1 ». Cette partie du mot logique est appliquée à une matrice brochée avec des diodes de façon que, sous l'action de l'ordre « BRA 1 » à l'entrée, s'excite le fil de sortie N° 1m (si on applique « BRA 2 » ou « BRA 3 », le brochage assure l'excitation des fils de sortie 2m, ou 3m).

Si la comparaison de p et q donne effectivement le cas 1 ($p > q$), du signal existe sur la sortie 1c du circuit de comparaison, ce qui, avec le signal engendré par la matrice sur sa sortie 1m, assure l'ouverture du « ET » N° 1. Si la comparaison de p et q ne donne pas le cas 1, mais mettons le cas 3 ($p < q$), du signal apparaît en 3c, mais il n'y en a pas en 3m (puisqu'on a appliqué l'ordre « BRA 1 ») et le « ET » N° 3 reste fermé. Si le cas 3 prévaut au lieu du cas 1 sur lequel on a misé, le « ET » N° 1 reste également bloqué, car il voit du signal en 1m mais pas en 1c. Il en est de même du « ET » N° 2, qui n'a de signal sur aucune de ses entrées. Il n'y a donc du signal sur la sortie du « ET » N° 1 *que* si le cas 1 est vérifié. Par le « OU » marqué H, ce signal parvient sur l'une des entrées du « ET » marqué C. Ce « ET » s'ouvre, car l'inverseur I applique aussi du signal sur son autre entrée puisqu'il n'y en a pas sur le fil t qui entre dans l'inverseur. La sortie de C excite l'une des entrées du « OU » marqué B, ce qui suffit, même sans signal sur t, à transmettre du signal sur l'entrée haute du « ET » marqué A. Dès lors cet « ET » laisse passer vers sa sortie S le signal qu'il reçoit sur son entrée basse.

Et ce dernier n'est autre que le *mot numérique j*, numéro du pas de programme qu'il faut transmettre au vade-mecum si le cas 1 s'est réalisé. En fait, ce signal est un mot entier fait de plusieurs tops et tips portés par plusieurs fils, de sorte que l'entrée basse de A et sa sortie sont en réalité multiples, c'est pourquoi on les a représentés par des traits plus gros. Si le cas 1 sur lequel on a misé en affichant l'ordre « BRA 1 » ne se réalise pas, il n'y a pas de signal sur 1m, pas de signal sur l'entrée



de H, et finalement pas de signal sur la sortie générale *S* : le N° *j* ne se transmet pas au vade-mecum, et celui-ci continue à dévider le programme dans son déroulement normal. On a bien obtenu un saut conditionnel dans le programme, c'est-à-dire un saut qui se produit si, et seulement si un certain test réussit.

On étend facilement le circuit pour comprendre aussi le saut *in*conditionnel vers le pas N° *j* : il suffit de brocher la matrice de façon que le fil *t* soit excité quand on affiche l'ordre « BRA $t \rightarrow j$ ». L'inverseur I, recevant alors un 1, engendre un 0 qui bloque le « ET » marqué C, et par suite interdit le passage à tout ce qui peut venir du comparateur. Par contre, le signal *t* lui-même passe par le « OU » marqué B, auquel cette entrée suffit pour aller ouvrir au nombre *j* la porte A. Vous savez maintenant comment on harnache l'ordinateur pour qu'il puisse obéir à des sauts conditionnels ou inconditionnels dans son programme.

L'addition

Nous entrons ici dans l'un des tabernacles du temple ordinateur, celui où s'accomplit le miracle de circuits électroniques additionnant deux nombres. Pour comprendre ce miracle qui n'en est pas un, nous parcourrons successivement trois étapes : d'abord une « *demi* »-addition de *deux digits seuls*, qu'on appelle « demi » parce qu'on n'y tient pas compte d'une éventuelle retenue issue de l'addition de deux digits antérieurs; ensuite l'addition « *complète* » de deux digits toujours seuls mais avec prise en charge d'une éventuelle retenue antérieure; enfin l'addition complète de deux nombres complets comportant plusieurs digits.

• *Première étape* : je veux additionner deux digits p et q. Qu'est-ce au juste que je dois engendrer ? Une « table de vérité » me le dira. Les deux premières colonnes de cette table contiennent comme toujours toutes les combinaisons possibles de valeurs de p et q. La troisième colonne donne le digit égal dans chaque cas à leur *somme*, et la quatrième le digit que je dois dans chaque cas conserver comme report, à intégrer ultérieurement. Vous n'aurez aucune peine à établir cette table, comme suit :

TABLE DE L'ADDITION
$p + q$

p	q	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(raisonner tout bas :
0 + 0 = 0 reste 0
1 + 0 ou 0 + 1 = 1 reste 0
1 + 1 = 2 qui s'écrit en binaire
10 c'est-à-dire 0 dans le rang
des digits qu'on additionne
et 1 dans le rang supérieur
ou report)

La colonne « R » montre que le *report* R n'est autre que la combinaison de p et q par l'opérateur logique « ET ». On sait donc former R d'emblée à partir de p et q par un circuit « ET ». Quant à S, vous constaterez que c'est une variable qui ne vaut 1 que *si une et une seule* des deux variables p et q vaut 1. Cette combinaison s'appelle un « **dilemme** » et se réalise facilement par le circuit dessiné dans la partie gauche de la figure. Il suffit d'énoncer le dilemme comme suit :

on a une sortie égale à 1

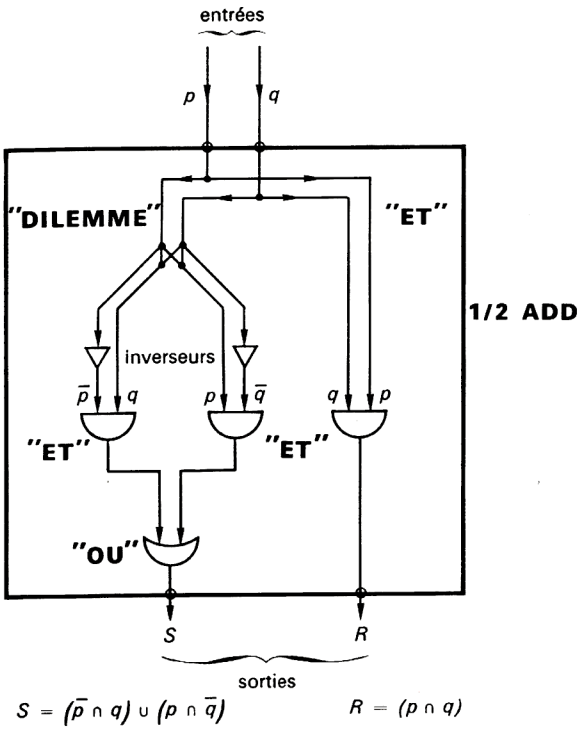
• ou bien **si**
$p = 1$ et $q = 0$,
c'est-à-dire $p = 1$ **et** $\bar{q} = 1$;

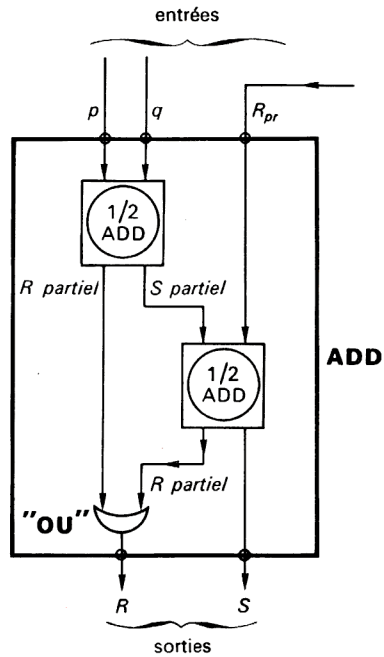
• ou bien **si**
$q = 1$ et $p = 0$,
c'est-à-dire $q = 1$ **et** $\bar{p} = 1$.

Le signal S doit donc émerger d'un circuit « OU » alimenté par les sorties de deux circuits « ET », l'un qui reçoit les signaux p et $\bar{q}$, ce dernier obtenu par inversion de q, et l'autre qui reçoit les signaux q et $\bar{p}$, ce dernier obtenu par inversion de p. C'est bien cette opération qu'accomplit l'ensemble des circuits de la figure dessiné à gauche des entrées p et q. En algèbre booléenne, on écrirait ces opérations :

$$S = \text{dilemme } (p, q) = (\bar{p} \cap q) \cup (p \cap \bar{q}),$$
$$R = \text{report} \qquad = (p \cap q).$$

Si j'entoure toute la circuiterie de la figure d'un carré, et que je n'y dessine plus les circuits logiques mais que j'inscrive simplement dessus l'étiquette « $\frac{1}{2}$ ADD » je peux considérer que cette boîte noire résout mon problème : à savoir que si on

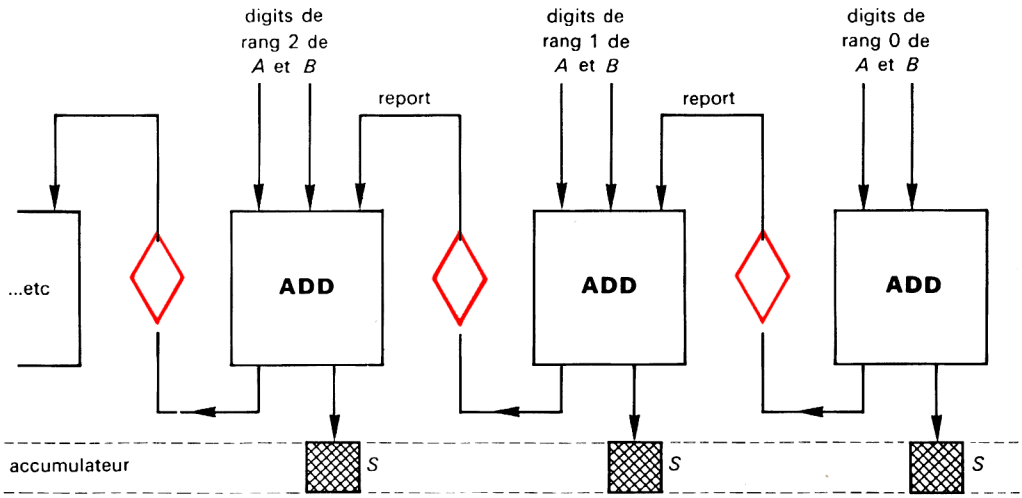




applique à ses deux entrées deux digits p et q, ses deux sorties fourniront la somme S et le report R de l'addition $p + q$.

• *Deuxième étape* : rappelez-vous que ce qui précède n'est qu'une « demi »-addition parce qu'on n'a pas encore introduit ce troisième larron que constitue le rapport éventuel issu de l'addition de deux digits occupant des rangs antérieurs. Demandons-nous maintenant ce qu'il faut faire d'un tel report R_{pr} (indice « *pr* » de « précédent »). Il faut évidemment l'ajouter au signal constituant la « somme » S (partielle) issue d'une première demi-addition des deux digits p et q. Ceci exige donc un second demi-additionneur admettant S partiel et R_{pr} comme entrées. Le signal « somme » qui en sort est bon comme « somme finale » de l'addition $p + q + R_{pr}$. Quant au signal « report » qui en sort également, il faut le mélanger dans un circuit « OU » au signal de report R partiel qui sortait du premier demi-additionneur, car le report final existe si l'un *ou* l'autre de ces deux reports partiels existent. On a ainsi bâti une boîte où s'additionnent deux digits p et q *plus* un éventuel report R_{pr} issu de l'addition de deux digits précédents. Je désignerai une telle boîte par l'étiquette « ADD ».

• *Troisième étape* : pour faire finalement la somme complète de deux nombres A et B doués de beaucoup de digits, il suffit de disposer d'autant de boîtes « ADD » qu'il y a de places dans les nombres. Les digits de même rang dans A et B sont appliqués à la boîte correspondante, qui reçoit aussi le report du rang précédent. Les digits des divers rangs dont l'ensemble constitue le nombre somme se forment sur les sorties « S » de ces boîtes.



Ils sont d'habitude admis dans ce registre spécial que nous avons appelé plus haut l' « *accumulateur* ».

C'est très simple. Il y a cependant deux petites anicroches. La première est que le fonctionnement des circuits de commutation a beau être ultra-rapide, il dure quand même un certain temps *non nul*. Or l'additionneur qui travaille dans un certain rang doit être informé du résultat obtenu par l'additionneur du rang précédent, puisqu'il doit traiter le report qui peut en être issu. Cela oblige à insérer entre les additionneurs de très courts retards entre rangs — c'est ce qu'illustrent les losanges de la figure — qui doivent être exactement dosés. D'où une série de complications dans la circuiterie, qui ne sont pas bien méchantes mais que je ne vous détaillerai pas.

La seconde anicroche vient de ce que deux nombres ne se présentent pas d'emblée sous la forme de deux mots de même longueur. Rappelez-vous qu'en dehors du « moule » des nombres qui remplit ces mots, il y a encore un « exposant », puissance entière positive et négative de la base du système de numération, qui fixe leur ordre de grandeur. Cela oblige encore à quelques manipulations en plus de celles représentées sur la figure. Considérez par exemple —, écrits en décimal — les deux nombres

$$A = 0{,}238\,74 \cdot 10^4 = 2\,387{,}4,$$
$$B = 0{,}186\,9 \cdot 10^2 = 18{,}69.$$

Vous savez bien qu'avant de pouvoir faire marcher la manivelle de l'addition, il faut « poser » un nombre par rapport à l'autre de façon que les virgules soient à la même verticale. Pour ce faire, on compare les deux exposants, en l'espèce 4 et 2, et on *décale* le nombre qui a l'exposant le plus petit vers la droite d'autant de rangs qu'il y a d'unités dans la différence. Ce n'est qu'alors qu'on procède à l'addition, et on donne à la somme l'exposant le plus fort, *plus 1* s'il y a un report au dernier rang. Toutes ces manipulations peuvent se faire sans trop de peine en *ordino* par des circuits logiques dans le détail desquels il est inutile d'entrer, et en outre un « *circuit de décalage* » que vous trouverez au chapitre suivant.

Philosophie

Voilà donc, cher lecteur, accompli le grand œuvre : nous disposons d'un circuit qui, lorsqu'on lui applique deux nombres (sous forme de suites de tops et tips) engendre leur somme. A partir de là, le reste de l'arithmétique va de soi. Pour soustraire *B* de *A*, on ajoute à *A* l'opposé de *B*, lequel se forme comme vous savez au moyen d'inverseurs. Pour multiplier *A* par *B* on ajoute *A* fois *B* à lui-même, opération qu'on peut exécuter de façon très ramassée dans un circuit qui, parce qu'il n'est pas de commutation pure et fait appel à des éléments de mémoire, sera traité au chapitre suivant. Pour diviser *A* par *B*, on met en œuvre un algorithme programmé d'avance et qui reproduit celui que vous appliquez à la main où ne figurent que des multiplications et soustractions. On peut donc dire que l'ordinateur sait exécuter toutes les opérations de l'arithmétique.

Et il sait le faire dans des temps fabuleusement courts. Observez, s'il vous plaît, que le circuit d'addition est bien « parallèle », comme je vous l'avais promis, c'est-à-dire qu'il admet et traite tous les digits *à la fois*. Ces digits le pénètrent par autant de fils en parallèle qu'il y a de places dans les mots numériques. En langage *ordino*, on appelle « ***temps machine*** » la durée d'*une* impulsion élémentaire et de la marge de sécurité dont on l'entoure. *C'est le « temps » fondamental d'un ordinateur*. Dans les machines modernes, il vaut de l'ordre de *1 microseconde*. Certaines machines ultra rapides descendent jusqu'à 0,1 μ s, d'autres plus « lentes » montent jusqu'à 10 μ s. Du fait que l'addition s'effectue en parallèle, elle ne dure *qu'un seul temps machine* (on s'arrange pour exécuter dans le même temps les autres manipulations auxiliaires dont je vous ai parlé). L'addition aurait au contraire duré *N* temps machine si on ajoutait des nombres de *N* digits *en série*, d'abord les digits d'un rang puis ceux du rang suivant, etc., le tout à travers un seul additionneur. C'est ce qu'on faisait au début, quand les circuits étaient chers et que le temps n'était pas critique. On peut donc aujourd'hui exécuter *un million d'additions par seconde*, quelles que soient les tailles des nombres. C'est dix millions de fois plus vite qu'à la main! Jamais encore le progrès n'avait exécuté un tel bond. Je vous l'ai déjà dit, je vous le redis, et je vous le répéterai encore : c'est cette fantastique ***vitesse***, que notre intuition est entièrement inapte à saisir, rendue possible du fait qu'on met en œuvre des circuits *électroniques*, qui constitue la nouveauté bouleversante des ordinateurs bien plus que l'astuce de la circuiterie.

Celle-ci cependant est digne de remarque, elle aussi. Vous avez sûrement admiré la belle progression qui conduit des circuits de commutation élémentaires, par un foisonnement ordonné de combinaisons, extensions et articulations, jusqu'à l'additionneur complet. Le principe sous-jacent de toute cette belle architecture **c'est que les opérations arithmétiques ont été ramenées à des opérations logiques** — lesquelles à leur tour se ramènent à des aiguillages et commutations. Or, cette réduction de l'arithmétique à la logique, quel est le secret qui la rend possible? Vous l'apercevez sans peine : *c'est la numération binaire*, qui permet de représenter un nombre par une suite d'existences ou non-existences, également interprétables comme des « ouis » et des « nons », ou encore des « vrais » et des « faux ». Or ces deux états, avec leurs combinaisons par « ET », « OU » et « NÉGATION », constituent aussi les briques de tout édifice logique. Nous parvenons ainsi à une vue très importante.

Je vous avais dit jusqu'ici que la raison du succès de la numération binaire dans les ordinateurs, c'est qu'elle assure la *simplicité* des opérations, et surtout la *sécurité* des interprétations puisque le « tout ou rien » ne laisse place à aucune ambiguïté. Or nous rencontrons maintenant une seconde raison, peut-être plus essentielle que la précédente : à savoir qu'en binaire, une addition apparaît comme une combinaison du « DILEMME » et du « ET », et peut donc s'exécuter en mettant en œuvre des circuits *logiques*. Cette réduction d'un *jeu sur les quantités* à un *jeu sur les endroits* est un secret très important (et insuffisamment publié) de l'univers *ordino*. Il enferme un enseignement qui dépasse cet univers. C'est que le mode de représentation des choses ou des idées n'est pas toujours un pur symbolisme. Écrire 101 pour désigner le nombre 5, en binaire, c'est faire plus que de nous permettre de parler commodément de cinq bâtons avec des signes au lieu des bâtons. Nous venons en effet de bâtir des opérations physiques sur la *nature* même de ces symboles, nous avons en quelque sorte utilisé leurs jambages comme piliers pour construire d'autres nombres significativement liés aux premiers. Les symboles nous ont servi non seulement de *signes*, mais encore d'*outils*. C'est le destin le plus haut qu'ils puissent ambitionner. Quand ils peuvent y accéder, c'est que nous avons particulièrement bien travaillé en les forgeant. Les symboles oraux et scripturaux par lesquels la langue courante désigne les choses et les idées n'ont pas encore atteint ce stade ultime de fertilité que nous donnent les symboles 0 et 1 des codages binaires.

Ce chapitre était destiné à vous faire voir que les impulsions électriques — dont les assemblages constituent les informations codées — peuvent être facilement soumises à toutes sortes de manœuvres telles que les grouper, sélectionner, aiguiller, conditionner les unes les autres, manœuvres qu'on réunit sous le vocable de *« commutations »*. Vous avez constaté que tout ça est possible grâce à la vertu qu'ont les *transistors* et des *diodes* électroniques de transmettre ou bloquer les impulsions, selon les signaux de contrôle qu'on leur applique, à des cadences fantastiques, comme par exemple 2 millions de fois par seconde. Vous avez fait connaissance ensuite avec un circuit de base à deux entrées appelé ***« ET »*** (ou encore *« circuit de coïncidence »*) qui donne du signal sur sa sortie si, et seulement s'il y a du signal à la fois sur les deux entrées. Puis, avec un autre circuit de base, également à deux entrées, appelé ***« OU »*** (ou encore *« circuit de mélange »* ou *« mélangeur »*) **qui** donne du signal sur sa sortie dès qu'il y a du signal sur l'une ou l'autre des entrées (ou sur les deux). Pour justifier l'emploi de ces conjonctions ***« et », « ou »*** on vous a entraîné un peu dans l'*« algèbre de Boole »* (où $1 + 1 = 1$ et $0 \cdot 1 = 1$). Après cette récréation vous avez appris à connaître l'architecture de quelques *« circuits logiques »* remarquables qu'on peut bâtir avec ces briques de base, tels que la *matrice d'aiguillage* qui permet d'exciter un fil ou un autre selon un numéro qu'on affiche, le *comparateur* qui permet de déceler si deux mots binaires sont identiques ou non, le circuit qui permet de réaliser un *branchement conditionnel* (ou inconditionnel), et enfin l'*additionneur* qui fournit la somme de deux nombres à l'instant même où on les applique au circuit. D'où on tire, pour clore le chapitre en beauté, quelques réflexions sur la fécondité du code binaire, dont la vertu cachée s'avère être qu'il ramène des opérations arithmétiques à des opérations logiques.

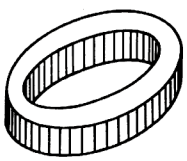
5

La mémorisation

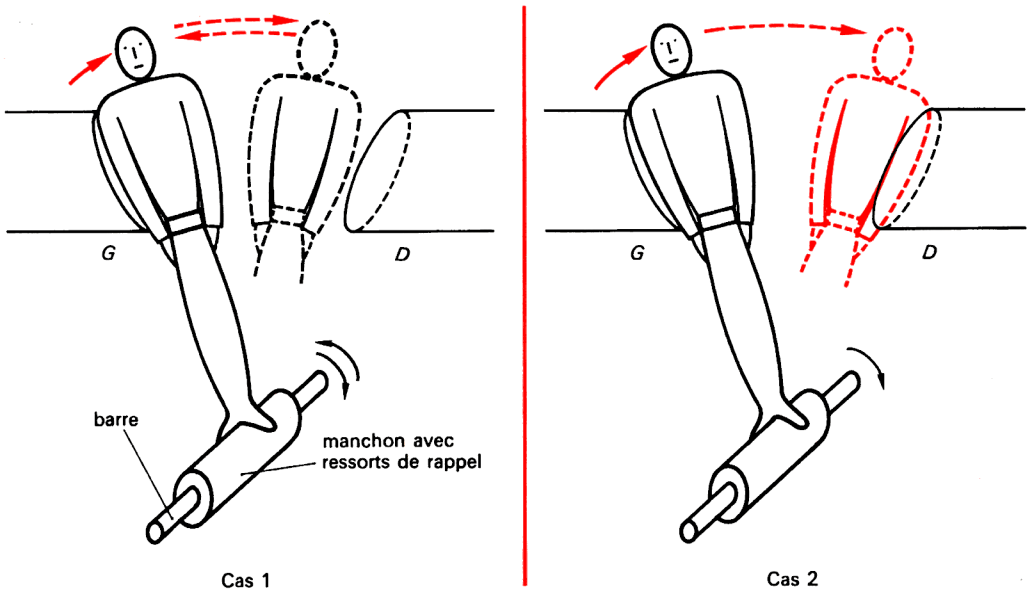
Dans tous les circuits de commutation que nous avons parcourus au chapitre précédent, les signaux étaient *fugitifs*. Ces circuits ne travaillent *qu'au vol*, au moment où des signaux les excitent. Nous allons maintenant faire connaissance avec des circuits qui, au contraire, *conservent*, par un changement *durable* de leur état, la marque des signaux qui les ont excités après que ceux-ci se soient éteints. C'est pourquoi on les appelle « circuits de mémoire ».

Le tore de ferrite, pendant magnétique du transistor

Le premier grain de matière miraculeuse qui a rendu possible l'ordinateur moderne a été ce cristal parfait exactement dopé dans lequel sont taillés les transistors. Vous avez vu qu'il domine les circuits de *commutation*. Les circuits de *mémorisation*, qui forment la seconde partie essentielle d'un ordinateur, ont été rendus possibles grâce à un second grain de matière miraculeuse qu'on appelle une « **ferrite** ». C'est un comprimé de poudres d'oxydes de fer et d'autres métaux qu'on prépare sous la forme de tout petits anneaux qu'en langage savant on appelle des « tores », et qu'une brodeuse appellerait des « perles plates ». Les dimensions sont de 0,5 mm pour l'ouverture intérieure et 1,5 à 2 mm pour le diamètre hors-tout. On en tient mille dans la paume de la main. Cette matière présente des propriétés magnétiques très remarquables, que je vais vous illustrer par une analogie, mais sans pouvoir vous les justifier, car cela exigerait de la physique savante.

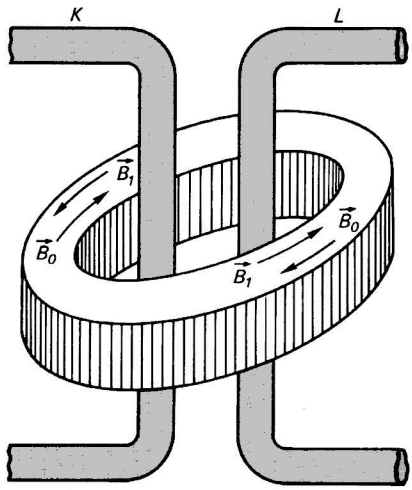


Considérez un petit soldat de plomb dont les pieds joints sont traversés par une barre horizontale. Autour des pieds se trouve un manchon à l'intérieur duquel des ressorts tirent dans deux sens opposés (ne cherchez pas à vous représenter exactement ce mécanisme). A gauche et à droite, on a disposé deux reposoirs, *G* et *D*. De ce fait, le petit soldat n'a que deux positions d'équilibre possibles, soit incliné sur le reposoir *G*, soit incliné sur le reposoir *D*. Supposons que des manœuvres antérieures l'aient laissé en position *G*, et administrons-lui sur la tête un coup perpendiculairement à la barre, de gauche à droite. Le bonhomme tend à tourner autour de la barre, mais si le coup n'est pas assez fort pour surmonter le jeu des ressorts de rappel (cas 1), ceux-ci ramènent le petit soldat à sa position d'équilibre stable *G*, et nous n'aurons eu qu'une oscillation avortée. Si au contraire le coup est assez fort (cas 2), le rappel exercé par les ressorts sera surmonté, et le petit soldat basculera



complètement vers sa seconde position d'équilibre, sur le reposoir de droite *D*. Si le soldat était initialement dans l'autre position d'équilibre, *D*, les deux mêmes cas — oscillation avortée avec retour vers *D*, ou basculement complet avec passage en *G* — se produisent sous un coup appliqué vers la gauche, selon qu'il est insuffisamment ou suffisamment fort. Bien entendu, rien ne bouge si l'équilibre initial est *G* et qu'on donne un coup vers la gauche, ou qu'il est *D* et qu'on donne un coup vers la droite.

Forts de ce modèle, tournons-nous vers le tore de ferrite, que je représente traversé par deux fils conducteurs *K* et *L*. L'aimantation d'une matière magnétique se représente par un *vecteur* que traditionnellement on désigne par la lettre « *B* ». Dans le cas où la matière est bien homogène et taillée en forme de tore, on démontre que le vecteur aimantation aux divers points s'aligne nécessairement le long des cercles concentriques qui courent dans le sein du tore perpendiculairement à son axe. Le vecteur $\vec{B}$ possède partout la même valeur (on dit que l'aimantation est « homogène ») et, dans l'état d'équilibre du petit anneau, il ne peut présenter que deux directions possibles, l'une tournant dans le sens des aiguilles d'une montre et l'autre en sens contraire. C'est le pendant des deux fameuses aimantations possibles d'un barreau magnétique qu'on appelle « nord » ou « sud ». Dans le cas de notre tore je me contenterai de désigner ces deux aimantations possibles par la même lettre $\vec{B}$ munie des deux indices « 0 » et « 1 », soit $\vec{B}_0$ et $\vec{B}_1$. Certains matériaux magnétiques sont dits « mous » parce qu'ils se désaimantent spontanément lorsqu'ils ne sont plus plongés dans un champ magnétique. La ferrite n'est pas de ce genre. C'est un matériau à magnétisme dit « dur », en ce sens qu'il persiste même quand le champ magnétique autour de lui disparaît. Les aimants du commerce sont de ce type. Une fois aimanté dans l'un ou l'autre sens, le tore laissé à lui-même conserve cette aimantation *indéfiniment* et ne peut pas en changer que sous une action que je vais vous décrire. Je dirai que lorsque son aimantation est $\vec{B}_0$ ou $\vec{B}_1$, respectivement, le tore « emmagasine »



(ou « enregistre », ou « conserve », ou « porte », ou « représente ») le digit 0 ou 1, respectivement.

Voyons cette action. Elle est basée sur les deux faits suivants bien connus :

— un courant électrique traversant un fil conducteur crée autour de lui un « champ magnétique » qui aimante toute matière aimantable se trouvant là, cette aimantation, pouvant disparaître ou persister après le passage du courant selon que la matière est à magnétisme « mou » ou « dur »;

— réciproquement, lorsque l'aimantation d'un morceau de matière magnétique *change*, il naît dans tout fil conducteur passant au voisinage, une tension électrique qui, elle, dans tous les cas, ne dure que *pendant* le changement même.

Appliquons le premier de ces deux faits au petit tore de ferrite. Nous supposons que des manœuvres préalables l'ont aimanté dans l'état B_1, où il subsiste indéfiniment si rien ne se passe autour de lui. Faisons se passer quelque chose, à savoir une impulsion de courant électrique dans *le fil K* qui traverse le tore par son trou central. Dans le cas de la figure, je fais passer ce courant impulsif de bas en haut (si j'avais désigné par B_1 l'autre état d'aimantation stable, il faudrait exciter le fil K de haut en bas). Le champ magnétique créé par ce courant tend à déplacer l'aimantation. Dans ces conditions, le tore ressemble au petit soldat de plomb initialement couché sur son reposoir de gauche et recevant un coup de gauche à droite. Si l'impulsion de courant n'est pas assez forte, l'aimantation de la ferrite n'est pas tirée assez loin hors de l'état B_1 et des forces de frottement magnétiques intérieures à la ferrite ramènent l'aimantation à l'état B_1, aussitôt l'impulsion de courant passée. Le magnétisme de la ferrite est tel que pendant cette tentative avortée de l'aimantation à quitter l'état B_1, *sa valeur ne subit qu'un changement insignifiant*. Il en résulte, d'après le deuxième fait rappelé tout à l'heure, qu'au total il n'y a pratiquement pas eu de changement d'aimantation, donc *pas de tension induite* dans le second fil métallique L qui traverse également le tore par son trou central.

Si au contraire, l'impulsion de courant dans le fil K (tout en conservant le même sens) possède une amplitude suffisante, alors la tendance de l'aimantation à changer

est assez forte pour vaincre l'inertie interne, et, tel le petit soldat de plomb qui bascule de G à D, l'aimantation subit une grosse modification qui la fait passer brusquement de l'état B_1 à l'état B_0. On dit en langage rapide que le tore « bascule », à l'instar de notre petit soldat de plomb. Comme ce basculement change l'aimantation de façon très appréciable, il induit dans le fil conducteur L une impulsion de tension. Ce « signal » apparaissant sur L est la manifestation palpable du fait que le tore était dans l'état 1. On peut dire que l'impulsion de tension induite par le basculement constitue une « lecture » du digit « 1 » qui était enregistré dans le tore. L'impulsion induite dans L ne dure, je vous l'ai dit, qu'autant que dure le basculement même. Ce dernier suit immédiatement l'impulsion de courant qui le provoque, commençant pratiquement avec elle et s'achevant un dixième de microseconde après. *Cette rapidité est à nouveau la clé de toute l'affaire.* Elle est due à ce que l'aimantation d'un morceau de matière magnétique est elle aussi liée, en fin de compte, à des arrangements spéciaux d'électrons dans les atomes, lesquels se font et se défont avec ce fantastique manque d'inertie qui caractérise les électrons. On a donc au total un organe qui peut stocker un digit et le débiter — ou comme on dit le faire « lire » — dans *un* seul « temps machine ». Un tel organe de mémorisation est pleinement compatible avec les organes de commutation travaillant à la microseconde que nous avons rencontrés au chapitre précédent.

L'organisation de la mémoire

▶▶ — Quelque chose me trouble, me dites-vous, ami lecteur. Quand le tore, ou plutôt son aimantation, bascule, pourquoi ce basculement n'engendre-t-il du signal que sur le fil de lecture L? Pourquoi pas aussi sur le fil d'excitation K?

▶ — Je vous demande pardon. Le basculement induit en effet une impulsion de tension aussi bien sur K que sur L. Mais sur K elle se noie dans l'impulsion d'excitation, laquelle est beaucoup plus intense.

▶▶ — Car la tension induite, elle, est faible?

▶ — Très faible. Au point qu'à la sortie du fil L il faut l'amplifier fortement pour obtenir du signal utilisable. Ce qui ne souffre pas d'objection à cet endroit, car sur L on ne rencontre pas de signal autre que cette impulsion induite par le basculement.

▶▶ — Autre chose encore me trouble. Vous m'avez expliqué comment on lit un « 1 » enregistré dans un tore, mais pas comment on lit un « 0 ».

▶ — C'est très simple si vous réfléchissez que la lecture d'un zéro, c'est-à-dire la manifestation du fait que le tore était aimanté dans l'état B_0, doit consister en ce qu'on récolte zéro, c'est-à-dire rien sur le fil de lecture L. Il suffit donc d'exciter le fil K avec *la même* impulsion de courant, de même sens que tout à l'heure, car vis-à-vis de B_0 ce sera un sens *antagoniste*, comme l'était une poussée vers la droite sur le bonhomme de plomb lorsqu'il reposait sur la butée de droite. Dans ces conditions, le tore, comme le bonhomme, *ne basculera pas* et on aura bien « 0 » sur le fil de lecture.

▶▶ — En somme, pour lire, on envoie dans tous les cas sur le fil d'excitation K

une *même* impulsion. S'il s'en forme aussi une dans le fil de lecture L, c'est que le tore avait stocké le digit 1. Et s'il n'apparaît rien, sur L, c'est que le tore avait stocké le digit 0.

▶ — Exactement.

▶▶ — Bon. Mais voici encore un autre trouble. Quand on lit un digit 1 qui était enregistré dans le tore sous la forme de l'aimantation B_1, ce tore ayant basculé, se retrouve avec l'aimantation B_0. On perd donc l'information par le fait même de la lire?

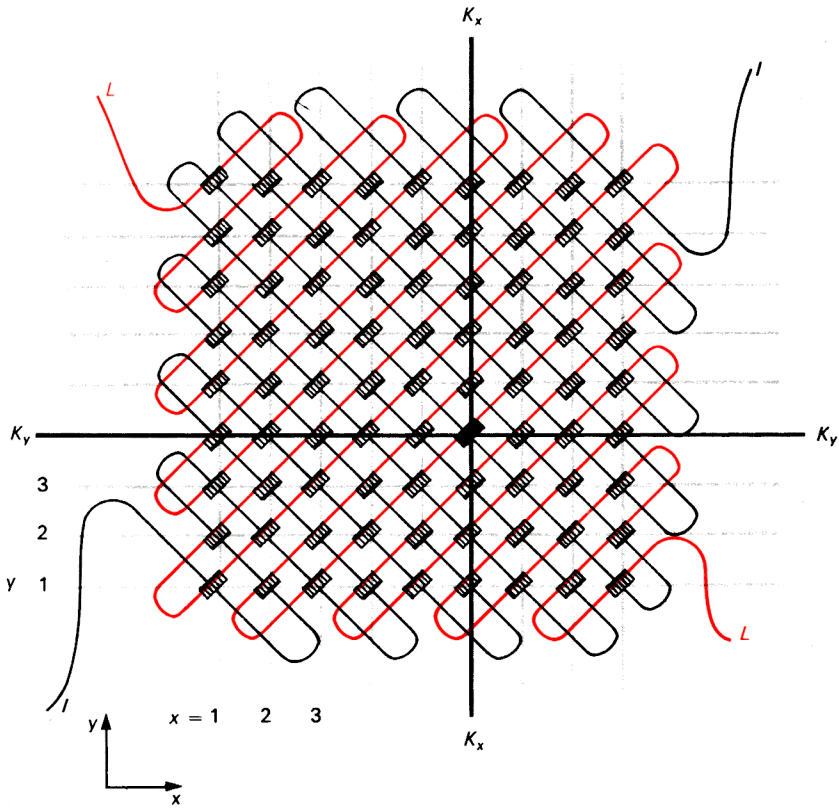
▶ — C'est parfaitement vrai, et ce caractère « destructif », comme on dit, de la lecture, constitue même le désavantage majeur de la mémoire à ferrites. On n'y peut rien. Le seul recours, quand on veut préserver l'information dans le tore. *c'est de la réinscrire aussitôt aprés l'avoir lue.*

▶▶ — Et comment *inscrit-on* un digit dans un tore? Vous ne me l'avez pas encore dit, et j'y vois une difficulté. Car si j'envoie dans votre fil d'excitation K une impulsion capable de faire basculer le tore, et que celui-ci est initialement dans l'état B_0, il passera dans l'état B_1. C'est bon si je voulais écrire un 1, mais c'est faux si je voulais inscrire un 0.

▶ — On pourrait croire que pour inscrire un 0 dans un tore qui se trouve dans l'état B_0, il suffit de le laisser en l'état, c'est-à-dire de ne rien faire du tout. Mais pour une raison que vous apprécierez dans un instant quand vous verrez une foule de tores organisés en une vaste mémoire, il faut qu'à chaque acte d'inscription (comme de lecture) on applique une impulsion sur K. Dès lors on s'en sort par l'astuce simple et jolie que voici. On fait passer dans le trou du tore un nouveau fil que j'appellerai I. Si on veut inscrire 1, on n'excite *pas* ce fil et on se contente d'exciter le fil K à pleine force, comme vous venez de le dire. Si on veut inscrire 0, on envoie toujours cette impulsion de pleine force dans K, mais en plus et en même temps, on envoie dans I une impulsion *de sens contraire* ayant la *demi-force*. Le tore subit alors au total une sollicitation de force $1 - \frac{1}{2} = \frac{1}{2}$, qui est insuffisante pour le faire basculer et le maintient donc dans l'état B_0. On dit qu'on a « inhibé » le basculement, et on appelle le fil I « fil d'inhibition ». Le résultat est qu'on a bien écrit un zéro dans le tore. Je dis « écrit » au lieu de « inscrit » parce que c'est la mode en langage *ordino*.

▶▶ — D'accord. Il reste une dernière question. Vous n'avez parlé jusqu'ici que d'un seul tore accommodant un seul digit. Comment fait-on pour mémoriser, écrire et lire, un mot entier, comportant plusieurs digits, et le situer à n'importe quelle adresse?

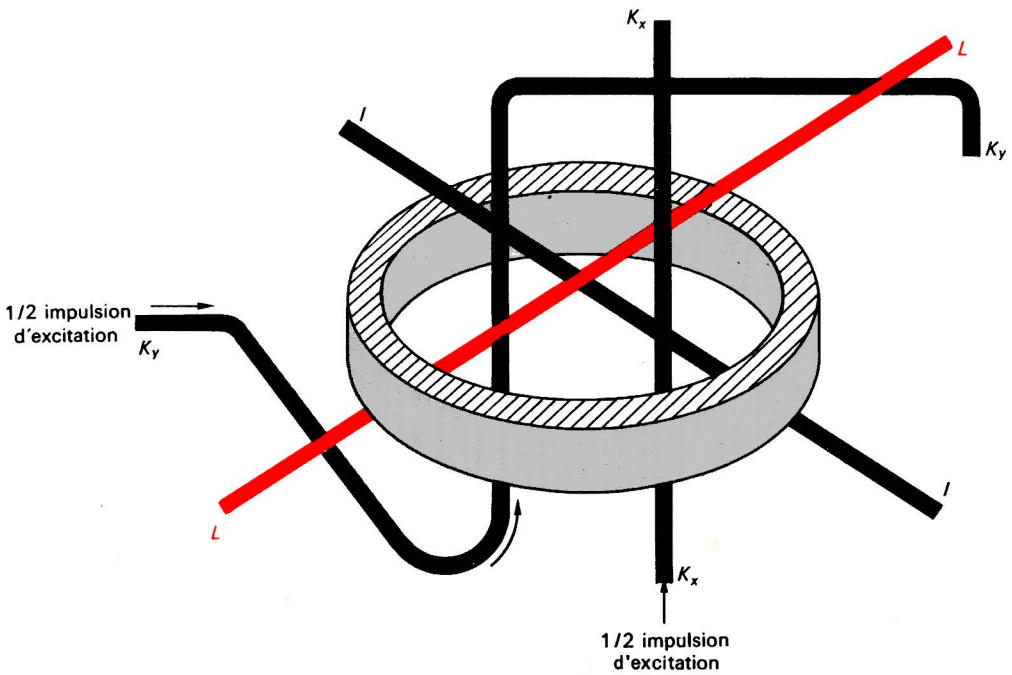
▶ — Question capitale. Pour la résoudre, on fait appel de nouveau aux propriétés magnétiques merveilleuses des ferrites. C'est maintenant que vous allez comprendre pourquoi les mémoires sont constituées par ces petites colonnes d'alvéoles dont je vous ai parlé dès le début. Rappelez-vous que leurs bases sont placées aux croisements d'une matrice carroyée par un réseau de lignes repérées par des abscisses X et un réseau de lignes perpendiculaires repérées par des ordonnées Y. Ne considérez pour l'instant que le plan de base, nous retrouverons dans un instant les autres plans. Sur ce plan de base on a disposé des tores aux croisements de la matrice. Chaque tore est traversé par *quatre* fils : deux d'entre eux passent dans *tous* les tores du plan,



ce sont les fils de lecture L et le fil d'inhibition I. Vous verrez qu'ils peuvent être communs à tous les tores parce que les choses seront arrangées de façon qu'il n'y ait jamais *qu'un seul tore du plan qui bascule à la fois.*

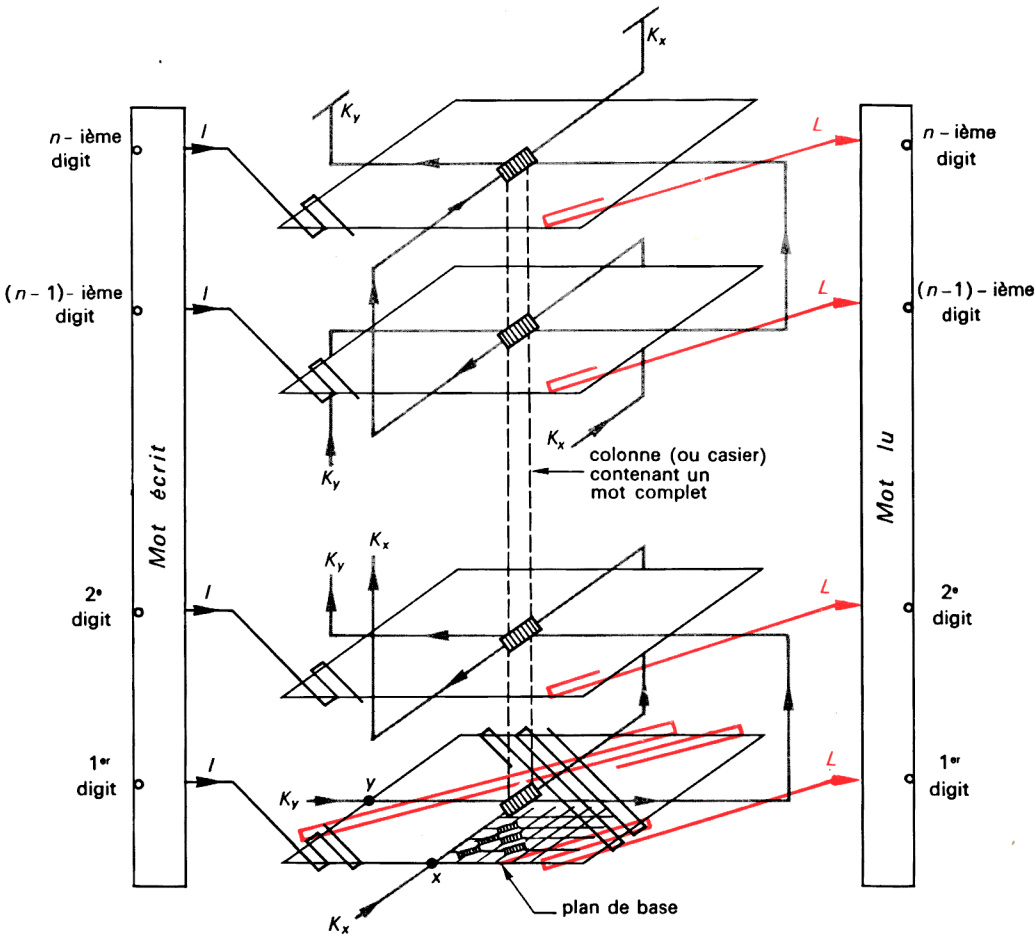
Quant à l'excitation, le truc génial consiste à la répartir sur ***deux fils séparés,*** dont l'un suit une verticale et l'autre une horizontale de la matrice. S'il y a N horizontales et N verticales on aura donc $N \times N = N^2$ tores traversés par N fils d'excitation verticaux K_x et N fils d'excitation horizontaux K_y.

Supposez maintenant que je veuille *lire* le tore situé à l'intersection de la verticale x et de l'horizontale y. J'envoie sur chacun des fils K_x et K_y, dans le bon sens, une impulsion de courant d'*une demi-force*. Tous les tores enfilés sur le même K_x et tous les tores enfilés sur le même K_y reçoivent cette demi-force, et par suite n'exécutent que des *tentatives avortées* de basculement; donc ils ne changent pas d'état et n'engendrent aucun signal sur le fil L. Tous les autres tores du plan, ne recevant pas du tout d'excitation, bougent encore moins et n'engendrent pas davantage de signal sur L. Vous voyez que, comme je vous l'avais annoncé, seul bascule le *tore visé*, car celui-ci coiffe *à la fois* K_x et K_y et reçoit donc $\frac{1}{2} + \frac{1}{2} = 1$ force entière. Du fait que ce tore est le seul actif, il est loisible de recueillir le signal qu'il engendre (1 ou 0) et qui me raconte dans quel état il se trouvait, sur un fil de lecture L traversant *tous* les tores du plan. Et si je veux *écrire* dans le tore situé à l'adresse (X, Y), c'est



très simple : après avoir mis tous les tores du plan à 0, j'envoie les deux *mêmes* excitations de demi-force sur les fils K_x et K_y et, *en plus*, *rien* ou une demi-force *négative* sur le fil d'inhibition I selon que je veux écrire un 1 ou un 0. Vous observerez que cette inhibition, n'ayant qu'une demi-force, ne peut par elle-même faire changer d'état à aucun tore.

Passons maintenant à des mots de plusieurs digits. L'idée est simple et vous en serez illuminé par un seul coup d'œil à la figure (voir page 116). On dispose autant de plans de tores identiques au précédent qu'il y a de places dans les mots à traiter, ces plans étant exactement empilés les uns sur les autres de façon que les tores des divers plans puissent se regrouper en colonnes. *Les tores d'une même colonne enregistrent les divers digits d'un même mot*, et ont tous la même adresse, celle du tore qui se trouve à la base. L'ensemble d'une colonne forme ce que nous avons appelé une « case » (ou un « casier ») de la mémoire, et accommode un mot complet. Tous les tores d'une même colonne, qui donc partagent une même adresse (X, Y), sont, comme vous le voyez, excités par *la même paire* de deux fils K_x et K_y qui serpentent d'un plan à l'autre. Dans chaque plan une ligne de lecture L unique est une ligne d'inhibition I unique, courent à travers tous les tores de ce plan. Il y a donc autant de fils L et de fils I séparés qu'il y a de plans, c'est-à-dire de places dans les mots. Si j'excite la paire de lignes K_x, K_y avec deux demi-forces, tous les tores de la colonne qui s'élève à l'adresse (x, y), et seulement ceux-là, sont sollicités pour un basculement complet; et chacun débite le digit qu'il a stocké (*un* ou *zéro*) sur la ligne L de son plan. L'ensemble des sorties des lignes L des divers plans fournit donc d'un seul coup la totalité du mot qui logeait à cette adresse. Si je veux *écrire* un mot à cette adresse, je commence par ramener tous les tores de la colonne à 0, puis j'envoie les mêmes



Le bloc mémoire complet.

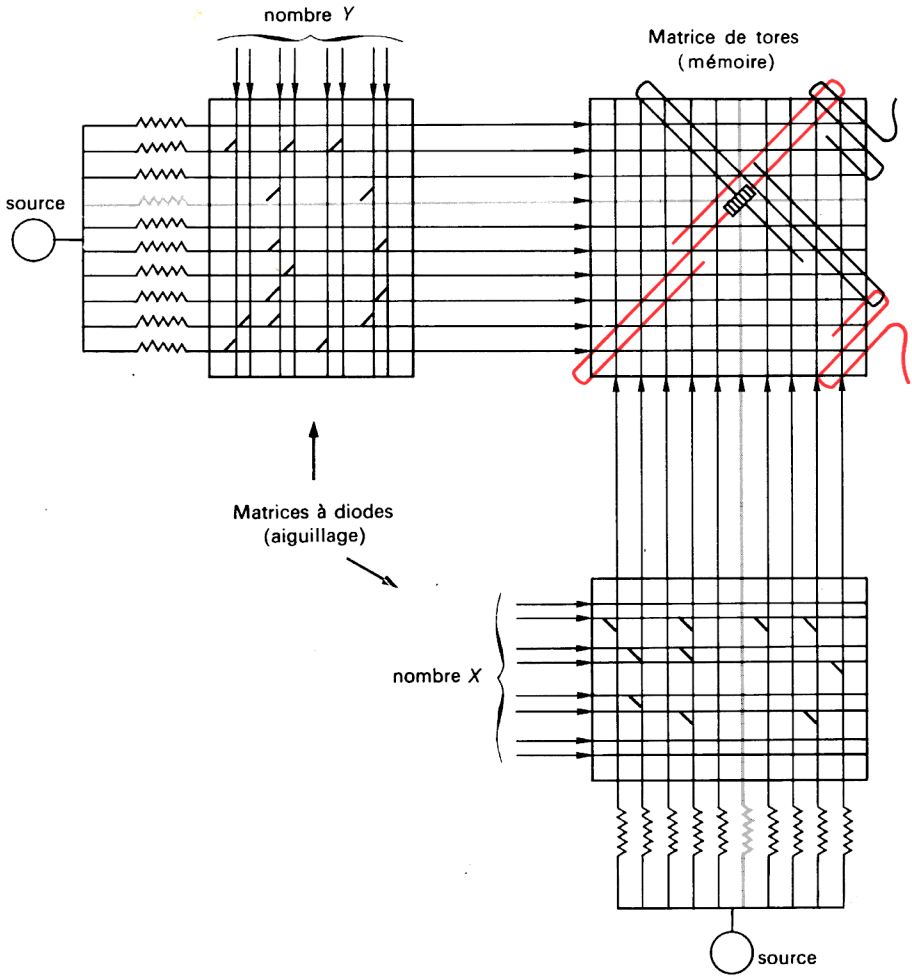
demi-forces dans les deux lignes K_x, K_y et, en plus, une demi-force inhibitrice dans les lignes I de ceux des plans où je veux écrire le digit 0. Et le tour est joué.

▶▶ — Et comment s'arrange-t-on, quand on ne dispose que de la paire de nombres X, Y formant l'adresse de la colonne à viser, pour que des courants d'excitation apparaissent aux pieds des deux fils K_x et K_y qui desservent cette adresse?

▶ — Vous ne devriez pas poser cette question. Car la réponse est contenue dans le chapitre précédent.

▶▶ — Vous voulez parler des matrices d'aiguillage peuplées de diodes aux croisements?

▶ — Bien entendu. Je vous avait dit que la fonction d'une telle matrice de diodes est précisément d'envoyer du signal sur un fil de sortie portant un certain numéro quand on applique ce numéro (sous forme de tops-tips) à son entrée. Donc nous allons connecter les fils K_x d'une part et les fils K_y d'autre part, aux sorties de deux



telles matrices dont les entrées reçoivent, pour l'une la partie X de l'adresse, pour l'autre la partie Y. Il suffit dès lors qu'aux deux entrées des deux matrices j'applique les tops-tips représentant les deux moitiés X et Y de l'adresse, pour qu'aussitôt du signal apparaisse sur les fils K_x et K_y, et que tout le mot situé à cette adresse s'inscrive — ou se débite — *dans l'instant même.*

Commentaires technologiques et admiratifs

Je suppose que vous avez dûment admiré l'astuce et la puissance des moyens mis en œuvre pour édifier une mémoire capable, *en un seul temps machine*, de faire *écrire ou lire* un mot situé en n'importe laquelle des adresses disponibles. Cette dernière vertu fait dire que la mémoire est à « *accès aléatoire* »; mais cette expression est malheureuse, il faudrait dire plutôt qu'elle est à « *sélection d'adresse discrétion-*

naire », comme le téléphone. Supposez par exemple que je dispose, dans *chaque* plan de 256 × 256 tores, soit d'un total de 65 536 tores, de sorte que j'aurai N fois 65 536 tores dans l'ensemble de la mémoire s'il y a N plans pour pouvoir traiter des mots de N digits. Ça fait déjà une belle mémoire, ces soixante-cinq mille et quelques mots de N places chacun! Or le nombre 256 est égal à 2^8, c'est-à-dire qu'une valeur quelconque de l'une des coordonnées X ou Y peut être formée avec un mot d'au plus *8* digits. Donc les deux matrices qui aiguillent des signaux vers les adresses voulues, offrent chacune 256 fils de sortie, mais n'ont besoin d'avoir que *8* entrées! L'énorme bloc des N plans contenant chacun 65 536 tores est ainsi contrôlé, en fin de compte, par seulement deux fois *8* paires de fils! Et ce contrôle très simple peut extraire ou introduire un mot de N digits à n'importe laquelle des 65 536 adresses en un seul temps machine, c'est-à-dire par exemple en 1 microseconde!

Cet exploit très remarquable ne s'obtient cependant pas sans effort. Les signaux d'excitation que les deux matrices aiguillent vers l'adresse voulue doivent être assez intenses et très rapides, ce qui exige une petite forge spéciale pour les préparer. De même pour les signaux d'inhibition. Quant aux signaux récoltés en lecture, je vous ai déjà dit qu'ils sont très faibles au sortir des tores, et en outre pas beaux, je veux dire qu'ils n'ont pas du tout l'allure idéale des impulsions rectangulaires. Il faut donc beaucoup les travailler au corps pour les amplifier, redresser et mouler. En outre, n'oubliez pas qu'après chaque lecture, il faut aussitôt réinscrire ce qu'on vient de lire, mais avec un petit retard pour ne pas empiéter sur les signaux d'excitation, et cela demande encore des circuits, toujours des circuits. Finalement, l'ensemble des plans de tores qui forme la mémoire proprement dite, disparaît sous un écheveau touffu de *circuits de service* jouant tous les emplois du théâtre *ordino*.

Et ce n'est pas tout. Les tores réels ne sont naturellement pas idéaux. Les petites oscillations avortées que provoquent les impulsions de demi-force sur les tores non voulus qui se trouvent alignés, en X ou en Y, avec le tore visé, ne sont pas *totalement* passives. Elles induisent aussi de faibles tensions parasites sur le fil de lecture. Très faibles certes, mais songez que le long de deux lignes K_x et K_y, qui se croisent sur le tore visé, sont suspendus 255 autres tores (dans un plan à 256 valeurs de X et de Y) qui tous vont exécuter ces petites oscillations avortées! L'addition des tensions parasites qu'elles induisent sur le fil L peut finir par dépasser le signal utile issu du basculement plein du tore visé. Pour contrecarrer cet effet, d'autant plus dangereux que la mémoire est plus grosse, ainsi qu'une foule d'autres petites interférences qui deviennent aussi inévitables quand on tasse beaucoup de tores les uns contre les autres que quand on tasse beaucoup d'hommes, il faut prendre une série importante de contre-mesures, comme par exemple d'inverser le sens de traversée des tores à chaque pas.

Le résultat est que le tissage de ces plans, avec les quatre fils qui traversent chacune des dizaines de milliers d'infimes perles plates de ferrite, est devenu un art dont la conception a mobilisé les meilleurs cerveaux de ce temps et dont l'exécution demande des ouvrières hautement spécialisées qui s'appellent très poétiquement « *brodeuses de mémoires* ».

Mais ce n'est pas encore tout. Le plus grave, c'est cette réflexion sur la fiabilité que je vous ai déjà présentée dès nos premiers pas dans l'univers *ordino* : une machine qui peut valoir 10 millions de francs est inutilisable si ***une seule*** de ces millions d'infimes

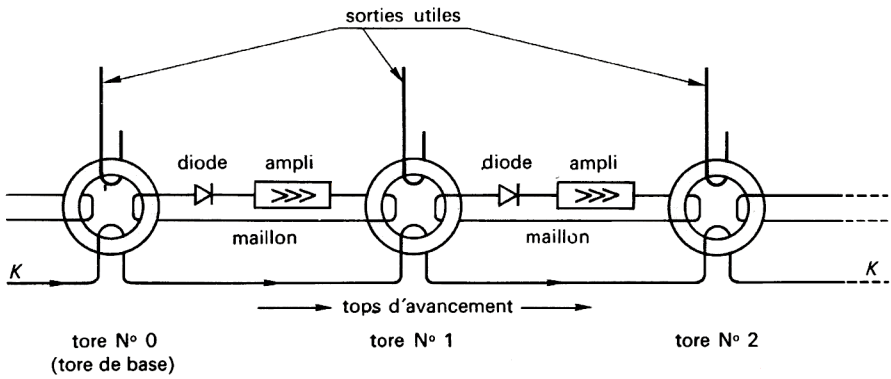
perles, valant peut-être 2 centimes, est déficiente. Jamais encore un projet n'avait exigé d'une industrie des performances aussi fantastiques. Puisqu'aucune faute n'est tolérable, il semble qu'on requiert la perfection, la vraie, celle qu'on dit inaccessible. En fait, deux circonstances viennent tempérer cet absolu. La première est que le grand principe de l'opération binaire, par « tout ou rien », conduit à n'exiger la perfection que d'un effet global et final qui, pendant sa genèse et avant son emploi, peut présenter des imperfections de détail sans compromettre la seule certitude qu'on demande en fin de compte, celle d'une existence ou inexistence. La seconde circonstance, c'est que la ferrite, comme les cristaux dopés qui constituent le substrat des transistors, est un matériau qui *ne vieillit pas*. Ce qui veut dire que si un tore a été reconnu bon une fois, on est sûr qu'il le restera toujours.

Dès lors, la réponse à notre problème de « fiabilité absolue » consiste à éprouver *chaque* tore, avant de l'envoyer au tissage, pour vérifier s'il présente correctement l'effet « tout ou rien » qu'on va exploiter. C'est plus vite dit que fait, car il est hors de question de contrôler *à la main* les millions de tores qui entrent dans un gros ordinateur, d'autant moins qu'un tore est à peine palpable, léger comme un souffle et ne demandant qu'à virevoleter en tous sens. Il a fallu déployer des trésors d'ingéniosité mécanique pour monter des machines de contrôle où ces toutes petites perles, sortant en flots tourbillonnants des fours qui les fabriquent, parviennent à s'écouler en bon ordre, se mettent en position correcte, tombent une à une exactement sur une fine aiguille dressée qui va les traverser et véhiculer les courants servant aux essais, puis s'évacuent automatiquement vers les deux sorties « bonnes » et « mauvaises » selon le résultat de l'essai. Ces machines en action fournissent l'un des plus notables spectacles de technique et d'esthétique qu'on puisse voir. Les ordinateurs ne sont devenus des vrais produits industriels qu'à partir des deux moments où on a mis au point d'une part ce contrôle automatique des ferrites, d'autre part la fabrication de qualité des transistors. Par une heureuse et remarquable coïncidence, ces deux moments ont coïncidé, ils se sont placés dans les années 1956-1958.

De la ferrite en toutes choses

Les services de la ferrite dans le monde *ordino* sont loin de s'arrêter à la mémoire.

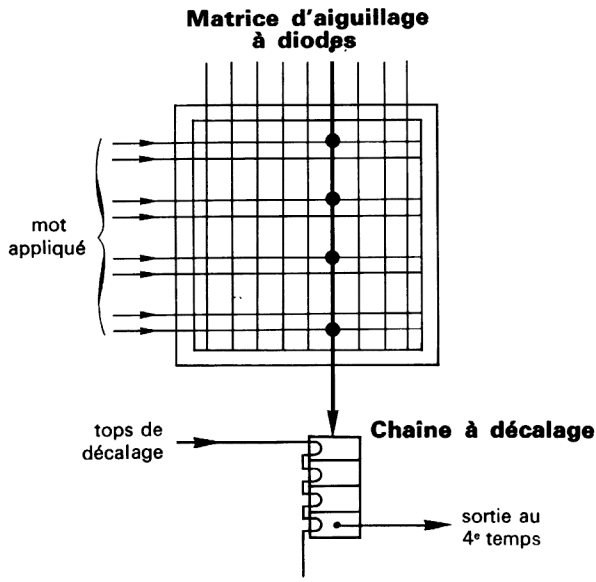
Considérons d'abord le problème de *distribuer* des tops, appliqués tous à une même borne d'entrée, sur une série de guichets de sortie disposés en file indienne. Je veux qu'au premier top appliqué, un top apparaisse au guichet de sortie N° 1, qu'au second top appliqué un top apparaisse au guichet de sortie N° 2, etc., le *n*-ième top appliqué se traduisant par l'apparition d'un top au guichet de sortie N° *n*. Vous sentez bien qu'un tel dispositif doit être utile dans le monde *ordino* pour envoyer des signaux aux endroits qu'on veut à des époques qu'on veut. Et je vous donnerai en effet une application précise dans un instant. Pour commencer, rêvons un peu à l'effet que je veux obtenir en imaginant que je suis un meneur de jeu assis à l'entrée d'un long couloir où donnent les chambres N^{os} 1, 2, 3, 4, 5, etc., à la queue leu leu. Dans chaque chambre il y a un diablotin qui m'obéit et je veux qu'à un premier ordre de moi (qui reste immobile) surgisse de sa porte N° 1 le diablotin N° 1, qu'à un second ordre surgisse de sa porte N° 2 le diablotin N° 2, etc. Un moyen simple serait de



téléphoner mon ordre chaque fois à nouveau, à chaque diablotin, à tour de rôle. Ce serait compliqué, et je devrai me rappeler où j'en suis. Il y a mieux à faire. Je m'arrange pour qu'un seul ordre, toujours le même (mettons, le mot « top »), entre, chaque fois qu'il est émis, dans toutes les chambres à la fois. Mais les diablotins ne l'entendent que s'ils sont correctement disposés, et je n'ai disposé correctement que le diablotin de tête. Quand le premier ordre est émis, il est seul à réagir, et j'ai mis un dispositif tel que *par le seul fait de sortir c'est lui qui dispose correctement le diablotin suivant*. Dès lors mon affaire se déroule beaucoup plus délicieusement. Il suffit que, paresseusement assis au début du couloir, je dise top, top, top, ..., au rythme que je veux, pour qu'au *n*-ième top ce soit bien le *n*-ième diablotin qui sorte de la *n*-ième chambre. Si j'ai fait sortir le 100-ième au 100-ième top, je peux attendre une heure ou une microseconde avant d'émettre le 101-ième, je suis sûr que dès qu'il sera émis le 101-ième diablotin sortira, car il avait été mis en condition par le 100-ième lorsque ce dernier était sorti.

En *ordino* les diablotins sont des signaux et leurs chambres des tores. Chaque tore est muni d'une boucle de sortie et tous reçoivent les tops récités à l'entrée par un fil *K* qui les enlace l'un après l'autre. Les connexions qui font qu'un diablotin, en sortant, conditionne pour sortir le suivant, sont les *maillons* dessinés, qui joignent un tore au suivant via une diode et un amplificateur qui ne laisse passer de signaux que dans le sens d'amont en aval. Le premier tore, qui sert de tête d'envoi est mis préalablement dans l'état 1, tandis que tous les autres sont, initialement, dans l'état 0. Je ne vous détaillerai pas le fonctionnement, mais en considérant avec attention la figure vous vous convaincrez vous-même sans peine qu'il est bien tel que je le souhaitais : chaque fois qu'un top est appliqué à l'entrée de la ligne *K*, un signal apparaît sur la boucle de sortie de l'un des tores parce qu'il bascule de 1 en 0. Et ce même basculement « positionne », c'est-à-dire, met aussi en 1, via le maillon allant vers l'aval, le tore suivant pour qu'il prenne le relais quand un nouveau top apparaîtra ; cependant que le maillon allant vers l'amont est bloqué par la diode, de sorte que les phénomènes ne peuvent qu'avancer et pas reculer.

Ce dispositif s'appelle une « **chaîne à décalage** », et parfois simplement un « **distributeur** » (de tops). Vous vous rappelez que la matrice de diodes transforme un *nombre en un lieu*. Eh bien, si on considère que les tops d'avancement se suivent à une cadence régulière d'horloge et marquent en quelque sorte le temps qui passe,

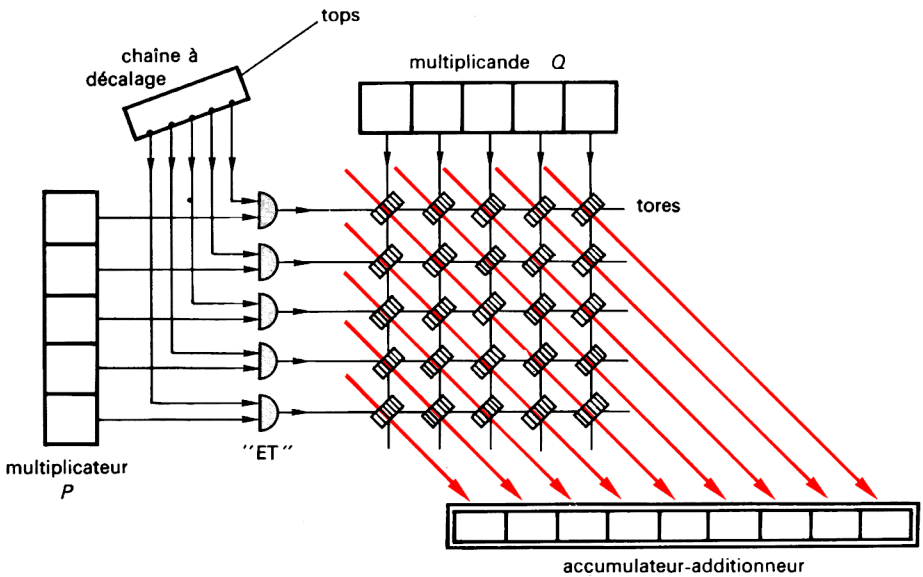


la chaîne à décalage apparaît comme un organe qui transforme *le temps en lieu.* Car à chacun des instants actifs je m'en vais exciter un autre fil de sortie.

Voici l'application importante de ce dispositif que je vous ai promise. Je viens de vous rappeler qu'une matrice d'aiguillage permet, grâce à un brochage adéquat des diodes, de faire surgir du signal sur tels de ses fils de sortie quand on lui applique un mot déterminé. Lorsque je vous ai présenté ce dispositif, je l'ai loué parce qu'il livrait le signal de sortie, sur le fil sélectionné, sans aucun retard, à l'instant même où on applique le mot. Mais dans certains cas, cette vertu est un défaut, car on peut vouloir disposer du signal à une époque ultérieure. Vous en verrez des exemples. Qu'à cela ne tienne. Il suffit d'admettre le signal immédiatement livré par la matrice sur la cellule de tête d'une chaîne à décalage afin de la mettre en 1. Puis on connecte la ligne K des tops de déplacement de cette chaîne à un générateur de tops qui, comme une horloge, délivre des tops à intervalles réguliers. Si on veut disposer du signal, mettons, quatre intervalles de temps *après* son émergence de la matrice, il suffit de mettre une boucle de sortie au *quatrième* étage de la chaîne.

On obtient ainsi un organe qui, sous l'action d'un mot donné, peut délivrer du signal non seulement sur les fils qu'on veut, mais encore à l'époque qu'on veut. C'est un organe qui assure un aiguillage sélectif à la fois dans l'espace et le temps, la sélection étant définie par la manière dont on a placé des diodes aux croisements de la matrice et des boucles de sortie aux tores des chaînes à décalage qu'on fait travailler à la sortie de la matrice. Une telle combinaison sera évidemment très puissante pour régler la marche des signaux dans un ordinateur en fonction de l'instruction qu'on veut faire exécuter.

Voici une autre application des ferrites et de la chaîne à décalages. Je vous avais dit au chapitre précédent que quand on veut multiplier un multiplicande Q par un multiplicateur P, on n'additionne pas bêtement P fois Q à lui-même. La figure ci-après illustre l'un des circuits astucieux qu'on peut mettre en œuvre grâce à l'emploi de



tores de ferrites. Le multiplicande Q est enregistré dans un registre, et de chaque case (chaque case héberge un digit) contenant un 1 sort en permanence un courant ayant la demi-force nécessaire au basculement, courant qui alimente les fils verticaux de la matrice de tores. De même, les digits 1 du multiplicateur P débitent en permanence des demi-forces sur les fils horizontaux de la matrice. Si ces derniers courants étaient admis sans autre forme de procès dans les tores, ceux-ci basculeraient tous. Au lieu de ça, on coupe la voie d'admission des digits de P par des « ET » dont les secondes entrées sont alimentées par les sorties successives d'une chaîne à décalage. Enfin des fils de lecture *diagonaux* et séparés mènent vers les cases d'un additionneur-accumulateur.

Vous devriez désormais être assez versé en circuiterie *ordino* pour comprendre ce qui se passe rien qu'en voyant la figure, mais je vais vous le dire quand même. Lorsqu'un signal — un top — émerge de la première sortie de la chaîne à décalage, le premier « ET » de la filière s'ouvre, et si le premier digit de P est un 1, tous les tores de la première ligne qui reçoivent un digit 1 du registre Q basculent et induisent ainsi du signal sur les cinq diagonales les plus à droite. Les zéros de Q donnent zéro. Ainsi, par le canal des fils de lecture diagonaux, le multiplicande Q se trouve recopié dans les cinq cases les plus à droite de l'additionneur-accumulateur. Si le premier digit du multiplicateur P était un 0, le multiplicande Q ne serait pas recopié. La seconde borne de sortie de la chaîne à décalage est connectée au second « ET » de la filière, auquel est connecté aussi le second digit de P. De la sorte, le second top issu de la chaîne à décalage provoque l'envoi du multiplicande Q vers l'additionneur si le second digit du multiplicateur P est 1, ou l'envoi de 0 si ce second digit est 0. Et *grâce à la disposition diagonale des fils de lecture*, cette seconde recopie de Q (si le second digit de P est 1) se place *automatiquement* d'un rang à gauche de la première, de sorte que son addition avec cette première est correctement cadrée puisque le second digit de P est d'un poids plus élevé. Ladite addition s'effectue automatiquement dès

que tombe la seconde recopie, décalée, de Q. Et ainsi de suite jusqu'à ce que tous les digits de P aient travaillé à leur tour et à leur rang, au bout de quoi l'accumulateur contient d'emblée le produit $P \times Q$. Vous voyez comme c'est élégant.

Pour votre troisième et dernière application des ferrites, vous aurez le droit de savoir comment on fabrique des sous-programmes « tous tissés », qui peuvent s'embrocher tout d'une pièce à un endroit préparé de l'ordinateur. L'épithète « sous », dans le mot « sous-programme », est seulement là pour signifier qu'on va insérer une suite d'instructions de structure toujours la même à un certain moment du déroulement d'un programme principal, mais à part ça un sous-programme ne diffère en rien d'un programme quelconque, c'est-à-dire qu'il est lui aussi une suite d'ordres. Mais rappelez-vous qu'en *ordino* un ordre est codé sous la forme d'un mot de tops et de tips. Un programme fait donc dérouler une correspondance déterminée entre les numéros des pas et les mots émis à chaque pas. Il faut que je m'arrange pour que, quand le vade-mecum récite le numéro n, il sorte un certain mot qui va aller travailler dans l'ordinateur, puis un certain autre mot quand le vade-mecum récite le numéro $n + 1$, etc. Pour fixer les idées, supposons que les ordres sont codés sous forme de mots logiques à 5 places, et mettons qu'aux pas N^{os} $(n - 1)$, (n) et $(n + 1)$ on doive faire apparaître les mots suivants :

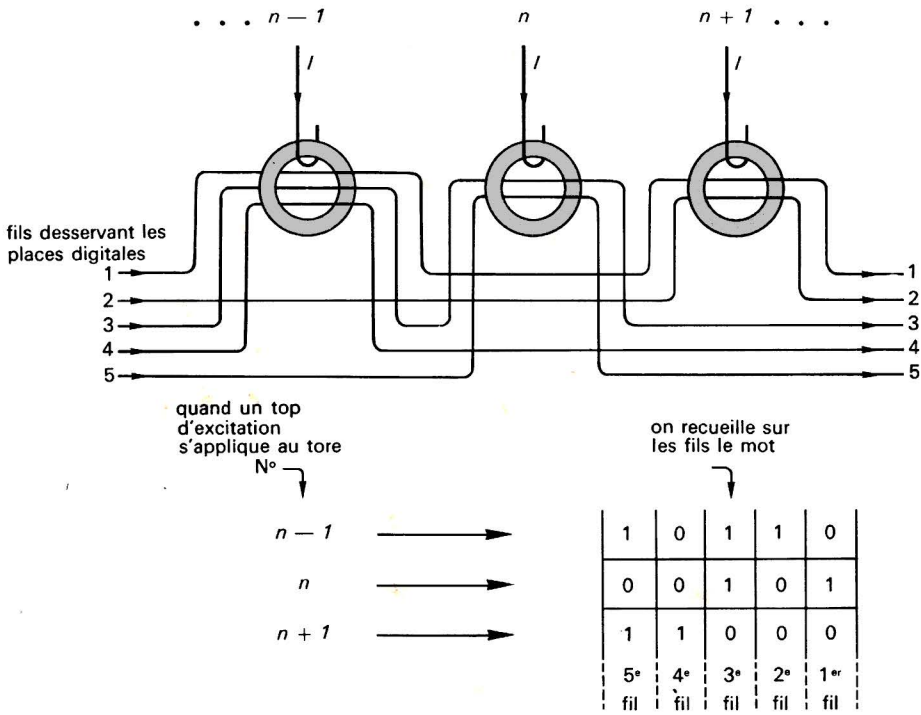
N° DU PAS	MOT LOGIQUE (ORDRE) CORRESPONDANT
$n - 1$	0 1 1 0 1
n	1 0 1 0 0
$n + 1$	0 0 1 1 1

Bien entendu, les mots vont s'incarner comme des tops et tips sur 5 fils, chacun des fils accommodant un digit. Il faut donc que, lorsque je récite « $n - 1$ », il apparaisse (je lis la première ligne du tableau) :

1 sur le 1er fil
0 sur le 2^{e} fil
1 sur le 3^{e} fil
1 sur le 4^{e} fil
0 sur le 5^{e} fil.

Quand je réciterai n, puis $n + 1$, il faudra que sur les 5 fils apparaissent les 1 et les 0 des deux autres lignes du tableau.

Or il y a une première chose que je sais faire, c'est « distribuer » des tops sur, des fils, c'est-à-dire faire en sorte que quand j'émets le $(n - 1)$-ième puis le n-ième puis le $(n + 1)$-ième top, il apparaisse du signal sur un fil N° $(n - 1)$, puis sur un fil N° (n), puis sur un fil N° $(n + 1)$. Dès lors je vais me servir de ces signaux émergeant sur des fils *côte à côte* pour exciter en 1 des tores placés *côte à côte* et numérotés $n - 1$, n, $n + 1$, et tous initialement en 0. D'autre part je saisis mon faisceau de 5 fils et j'exécute avec eux une petite prestidigitation chinoise d'enlacement et de non enlacement. Puisque je veux qu'en excitant le tore N° 1 le 1er fil porte un 1, le second un 0, etc. (voyez le tableau), c'est très simple : je fais passer le 1er fil par ce tore, le second fil pas, le 3^{e} fil oui, le 4^{e} fil aussi, le 5^{e} fil non. Et quand le tore N° $n - 1$ est excité, au pas N° $(n - 1)$, c'est-à-dire qu'il bascule, il induit ainsi du signal sur le 1er fil, pas de signal sur le 2^{e}, etc. C'est-à-dire que le mot qui apparaît sur le faisceau



de 5 fils est bien celui que je voulais au pas $n - 1$. Je fais de même avec les tores suivants N° n et N° $n + 1$, c'est-à-dire que je les fais traverser ou pas traverser par mes fils porteurs de digits selon que je veux qu'apparaisse sur eux un 1 ou un 0, aux pas n ou $n + 1$, quand les tores N° n et N° $n + 1$ basculeront. Et comme, dans le sous-programme, la suite des mots à engendrer est fixée *ne varietur*, j'ai un organe tout préparé qui me débitera cette suite chaque fois que je l'appellerai. Le tour est joué.

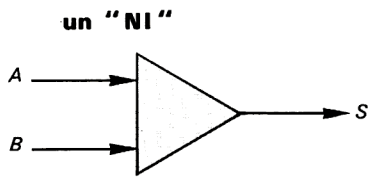
Et un tour beaucoup plus considérable est joué du même coup. Songez en effet que ce dispositif peut m'engendrer non seulement des programmes fixes tout tissés, mais encore n'importe quelle fonction mathématique $y = f(x)$! Rappelez-vous qu'une telle fonction fait correspondre une valeur bien déterminée d'une variable *dépendante* y à toute valeur d'une variable *indépendante* x. Eh bien, les diverses valeurs possibles de x seront les divers *rangs* dans ma filière de tores, et quand je me donne l'une de ces valeurs, je sais, par une matrice d'aiguillage à diodes, aller exciter le tore qui lui correspond, c'est-à-dire dont le rang est égal à cette valeur. Et j'aurai d'autre part tissé ce tore, avec les fils qui doivent porter les digits, de façon que lorsque ce tore bascule, la configuration de tops et tips que cela fait apparaître sur les fils forme la valeur de y que ma fonction demande de faire correspondre à la valeur de x que j'ai appliquée. C'est ainsi qu'une collection de tores traversés et non traversés de façon adéquate par des fils de lecture, constitue l'*incarnation d'une fonction.* On peut y lire en un seul temps machine quelle valeur de y la fonction fait correspondre à *n'importe quelle* valeur de x qu'on se donne. Et les fabricants peuvent vendre ces « *tables de fonction* » *toutes tissées*!

Les « flips-flops »

J'ai passé rapidement et pudiquement sur le défaut que présentent les mémoires magnétiques d'avoir une lecture destructive. Dans certains cas ce défaut est rédhibitoire, il faut pouvoir consulter et faire travailler l'information stockée sans qu'elle disparaisse; ou encore remplacer une information ancienne par une information fraîche simplement en *superposant* cette dernière. On doit avoir ces propriétés en particulier chez ces « registres » dont je vous ai parlé, qui mémorisent des informations dans des coins spécialisés de l'ordinateur pour des tâches locales et transitoires, registres qu'on appelle parfois des « *mémoires bloc-notes* ».

Ces petites mémoires isolées sont construites à partir d'un circuit fondamental de l'électronique qu'on appelle une « **bascule** », en anglais un « **flip-flop** », car chez lui aussi, comme chez un tore de ferrite, le phénomène clé sera un *basculement*. Mais cette fois il consistera non pas dans un changement brusque d'*état magnétique*, mais dans le changement brusque du *chemin* offert à un courant électrique.

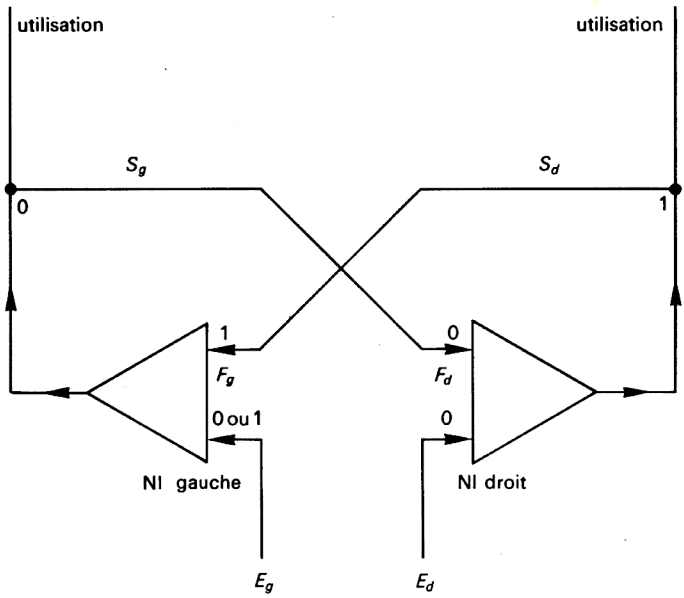
La clé de ces « bascules électroniques » est un circuit logique qu'on appelle « NI », qui possède deux entrées et une sortie, qu'on représente par un petit triangle comme sur la figure, et dont le fonctionnement consiste à livrer du signal sur la sortie *S si, et seulement si* il n'y a de signal **ni** sur l'entrée *A* **ni** sur l'entrée *B*. Toute autre combinaison aux entrées : — signal sur l'une mais pas sur l'autre, ou signal sur les deux — entraîne la nullité sur la sortie. On a démontré qu'on peut réaliser *toutes* les expressions booléennes remarquables uniquement par des combinaisons de circuits « NI ». Un « NI » peut s'obtenir soit en plaçant deux inverseurs sur les deux entrées d'un « ET », soit en plaçant un inverseur sur la sortie d'un « OU », soit encore mieux avec une seule triode à transistor dont on prend la « base » et l' « émetteur » pour entrées et le « collecteur » pour sortie, dans un arrangement que je ne vous détaillerai pas.



$S = 1$ si, et seulement si
$A = 0$
et
$B = 0$

On connecte maintenant deux circuits « NI », un « à gauche » l'autre « à droite », selon le pittoresque chassé-croisé de la figure de la page 126, où la sortie S_g du membre gauche retourne sur l'entrée haute F_d du membre droit, tandis que la sortie S_d du membre droit retourne sur l'entrée haute F_g du membre gauche. Les deux entrées basses des deux membres, c'est-à-dire E_g à gauche et E_d à droite, sont laissées libres pour servir d'entrées à l'ensemble.

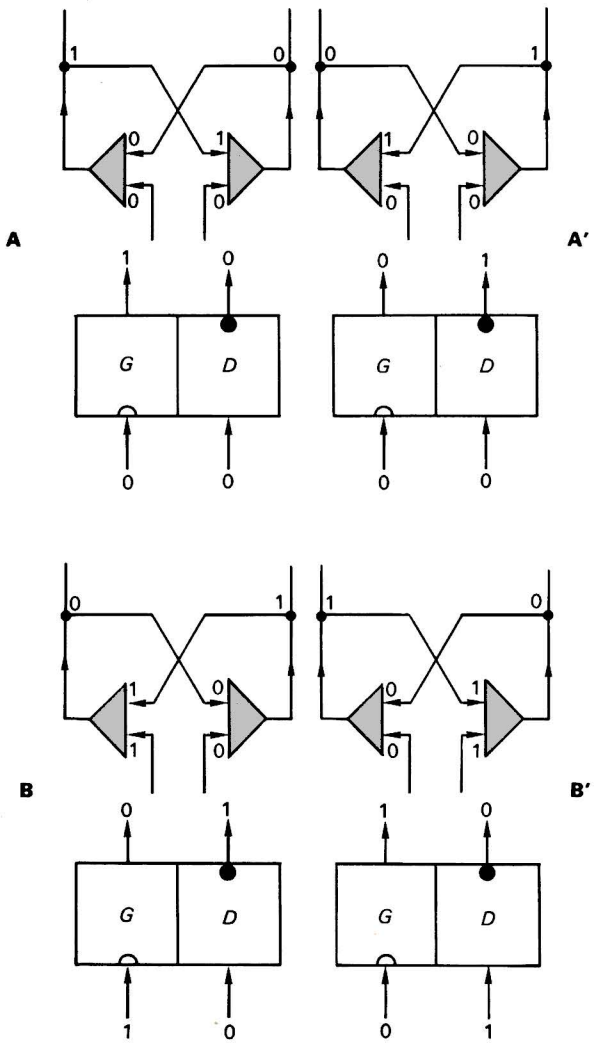
Il est facile de voir que ce système ne peut se maintenir équilibré que lorsqu'il y



a du signal sur une seule des deux sorties et pas de signal du tout aux entrées, ou du signal seulement sur l'entrée *opposée* à la sortie qui en a. Imaginez par exemple qu'*il y a du signal en* S_d, sortie de droite. Il y en a donc du même coup en F_g, entrée haute du « NI » de gauche. Dès lors ce « NI » de gauche impose qu'*il n'y a pas de signal en* S_g, comme annoncé. Donc il n'y en a pas non plus en F_d, entrée haute du « NI » de droite. Quant aux entrées, l'existence de signal à la sortie de droite impose qu'il n'y en ait pas sur l'entrée basse de droite, E_d, de même qu'il n'y en a pas sur l'entrée haute de droite, F_d. L'entrée basse de gauche E_g, qui est *en biais* par rapport à la sortie qui a du signal, peut, elle, avoir ou ne pas avoir de signal sans enfreindre les lois du « NI » de gauche.

Cette analyse très simple montre qu'il n'y a que quatre états d'équilibre stable possibles, représentés par la figure, où les carrés schématisent des circuits. Par « équilibre stable » on entend un état qui se perpétue tel quel tant qu'une action extérieure ne le perturbe pas. Si vous inspectez ces quatre états avec un tantinet d'attention vous constaterez que *les signaux aux deux sorties sont toujours complémentaires.*

Pour illustrer le mécanisme par une *évolution* d'états, supposons par exemple qu'on privilégie l'une des deux sorties, mettons celle de droite, pour en faire la sortie « *noble* » en ce sens que c'est là qu'on lira les signaux portant de l'information utile. C'est ce qui est marqué sur la figure par un gros point à la base de cette sortie. L'autre sortie pourra se connecter à des circuits ultérieurs pour accomplir des tâches « *de service* ». Prenons d'autre part pour entrée « *noble* », où seront appliqués les signaux porteurs d'information, celle marquée d'une demi-lune, qui est *en biais* par rapport à la sortie noble, l'autre entrée étant « *de service* ». Et partons de l'état *A* de la figure, avec la sortie noble en 0 et par suite la sortie de service en 1, les deux entrées étant en 0. Sur l'entrée noble j'applique du signal 1 que je vais d'abord supposer *durable*. Je



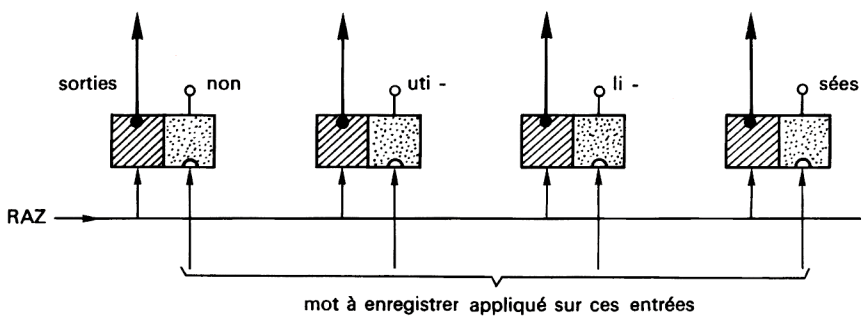
passe aussitôt à l'état *B* où les signaux sur les deux sorties se sont intervertis. *C'est cela qu'on appelle le* « ***basculement*** ». On a réussi à dimensionner des circuits dans lesquels ce basculement s'exécute en moins d'*un centième de microseconde*, c'est-à-dire en un temps négligeable par rapport à la durée des tops les plus fugaces qu'on rencontre dans les ordinateurs. On peut donc dire que le basculement est pratiquement *instantané*. Le nouvel état *B*, où la sortie noble possède un 1, persiste comme persistait l'état précédent *A*. Il persiste même, et ce fait est fort intéressant, *si l'entrée noble rétrograde à 0*, car cela conduit à l'état *A'* qui présente la même configuration des sorties que l'état *B*. On en déduit qu'une fois que l'entrée noble a été excitée, même transitoirement, par un top qui après son passage laisse la bascule dans l'état *A'*, toute nouvelle excitation sur cette même entrée noble ne peut plus provoquer que des passages entre les états *A'* et *B*, ce qui *ne change plus* le 1 présent sur la sortie noble. *On a*

donc bien réalisé une mémoire indestructible, mémoire du fait que l'entrée noble est passée une fois de 0 à 1.

Plus exactement, on peut quand même détruire cet état 1 sur la sortie noble, et faire revenir celle-ci à 0 par migration de son 1 vers la sortie de service, mais pour cela il faut appliquer un 1 (passager ou durable au choix) sur l'entrée qui fait *vis-à-vis* à la sortie noble, c'est-à-dire l'entrée de service.

Ce qu'est un « registre »

Les applications des bascules électroniques sont aussi nombreuses que leurs anatomies. Je ne vous en citerai que deux. La première est la réalisation de l'objectif même qui nous a fait ouvrir le sujet des bascules, à savoir un « registre ». Il suffit de placer autant de bascules côte à côte qu'il y a de places dans le mot à mettre en registre. Sur les entrées « nobles », placées de la même façon dans toutes les bascules, on applique les digits du mot à enregistrer sous forme de tops-tips transitoires, et si les sorties nobles d'où émergent les signaux utilisés étaient toutes initialement en 0, elles prennent aussitôt les états 1 ou 0, correspondant aux divers digits du mot. Et ces états sont *permanents*. On peut à tout moment s'assurer que ces sorties nobles des bascules sont bien initialement à 0 en appliquant un top 1 simultanément sur toutes les entrées non exploitées pour les digits de l'information, c'est-à-dire les entrées que j'ai appelées « de service » et qui sont placées vis-à-vis des sorties nobles.



C'est à quoi sert la ligne marquée du curieux sigle « **RAZ** » qu'on prononce comme le début du verbe « raser ». Devinez-vous ce qu'il signifie ?

▶▶ — Non, je ne vois pas.

▶ — Tout simplement « Remise A Zéro ». C'est une fonction que vous rencontrez pour la première fois mais qui est fort importante dans une machine complète. Or les registres, qui ont sur les mémoires magnétiques l'avantage qu'on peut lire leur information sans qu'elle disparaisse, ont le désavantage qu'une information qu'on y inscrit ne chasse pas celle qui aurait pu y être déjà inscrite, mais se combine à elle pour donner des effets aberrants. C'est comme une armoire où on range des serviettes blanches identiques. Si on veut identifier correctement une série, il faut d'abord vider les rayons de toute autre série antérieure.

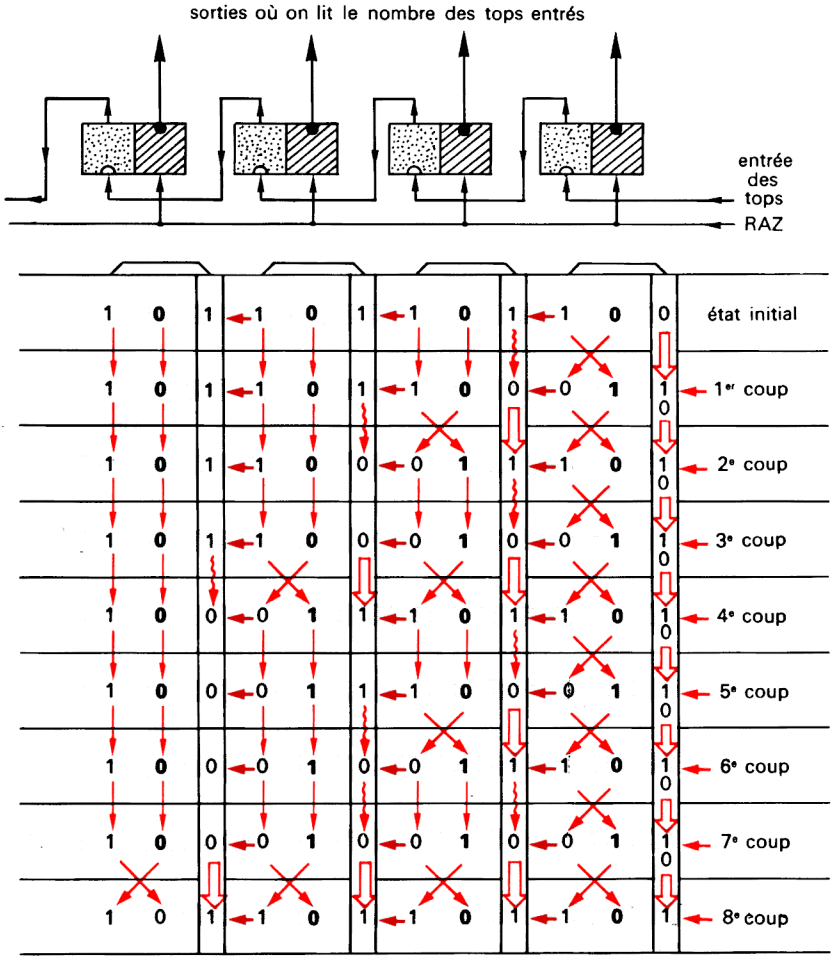
Qu'est-ce qu'un « compteur »?

Jusqu'ici nous avons vu des tops travailler dans toutes sortes de conditions, mais chacun pour son compte. Or il est souvent utile de savoir *combien* de tops sont passés à un endroit déterminé depuis un certain moment pris pour origine, et très souvent même leur *nombre* joue un rôle constructif. Ainsi, si on veut former la somme des 17 premiers termes d'une série de nombres dont les termes sont inscrits à des adresses consécutives de la mémoire à partir d'une adresse n_0 initiale, il faut pouvoir compter le nombre des tops de lecture qui vont chercher ces termes dans la mémoire pour les envoyer à l'additionneur, afin que la cueillette s'arrête quand on en a pris 17. Ce nombre des tops qui ont défilé à partir d'une origine, le compteur les présente à sa sortie sous forme *binaire*. En outre, on demande au compteur de faire son travail de dénombrement des tops quel que soit le rythme de succession de ceux-ci : chaque fois qu'il en défile un nouveau, que ce soit une microseconde ou une année après le précédent, le compteur doit augmenter de 1 le nombre qu'il affiche.

On va bien sûr utiliser, pour construire un compteur, des bascules, mais pas tout à fait élémentaires. La bascule élémentaire présente en effet cette particularité, qui est utile dans les registres mais gênante dans les compteurs, qu'une fois la sortie noble mise en 1 par un passage de 0 à 1 de l'entrée noble, celle-ci ne peut plus déloger le 1 quoi qu'on lui applique. Mais il n'est pas très difficile de modifier ce comportement en interposant un circuit logique tel que si la sortie noble est en 1 et qu'on réexcite l'entrée noble de 0 à 1 cette excitation se transpose automatiquement sur l'entrée de service, laquelle comme je vous l'ai dit assure le retour de la sortie noble à 0. Je ne vous encombrerai pas du schéma de ce circuit logique, d'ailleurs très simple, car avec tout ce que vous savez vous ne doutez pas que ce soit possible. Les bascules retouchées de cette façon peuvent s'appeler « *polarisées* » et je les distingue des autres par une marque sur le côté en faveur duquel on a rompu la symétrie. Une bascule polarisée *bascule à* ***chaque fois*** *que l'entrée noble passe de 0 à 1*, passage que nous allons appeler « *aller* », et ne bouge naturellement pas lorsque l'entrée noble passe de 1 à 0, passage que nous appellerons « *retour* ».

Ceci dit, je prends une série de bascules polarisées que je connecte entre elles comme il est montré sur la figure de la page 130. On commence par s'assurer, au moyen d'un RAZ appliqué partout, que toutes les bascules sont à 0 (sur leurs sorties nobles, donc à 1 sur leurs sorties de service). Puis on va appliquer à l'entrée noble la plus à droite, qui jouera le rôle d'entrée de toute la chaîne, les tops successifs à compter, et on va examiner les états successifs des bascules de la chaîne. Sur la figure, les états des entrées nobles sont inscrits dans les petites colonnes, où une flèche serpentante signifie un passage « retour » et une double flèche un passage « aller ». Les états des sorties nobles sont en chiffres gras. Les états des sorties de service sont liés par des traits aux états des entrées nobles suivantes, pour souligner qu'ils sont identiques puisque connectés entre eux.

Lorsqu'on applique le *premier* top, la bascule la plus à droite, voyant son entrée passer de 0 à 1, c'est-à-dire dans un sens « aller », bascule, faisant apparaître 1 sur sa sortie noble et 0 sur sa sortie de service. Donc, sur l'entrée noble de la 2^{e} bascule, il y a un passage « retour », $1 \rightarrow 0$, et cette 2^{e} bascule ne bascule pas, elle et tout le reste de la chaîne ne bougent pas. On a engendré l'état écrit dans la ligne marquée « 1er coup ». Puis on applique un *second* top. Ici, attention! Les excitations à l'entrée



de la chaîne sont de *vrais tops*, c'est-à-dire des impulsions *fugitives*. De sorte que le 2e coup retrouve l'entrée de la 1re bascule *à zéro*. Il n'en va pas de même des autres signaux le long de la chaîne, qui sont issus de bascules et donc persistent tant qu'on ne vient pas les déranger. La 1re bascule voit donc à nouveau sur son entrée une impulsion « aller » de 0 à 1 et bascule à nouveau, retrouvant ainsi sur ses sorties noble et de service les valeurs 0 et 1 du début. Le 1 de la sortie de service s'applique à l'entrée noble de la 2e bascule que le coup précédent avait laissée à 0. Cette 2e bascule reçoit donc cette fois, elle aussi, une excitation « aller », et bascule. Ce qui fait apparaître 1 sur sa sortie noble, 0 sur sa sortie de service et donc aussi sur l'entrée noble de la 3e bascule, que le coup précédent avait laissée en 1. Cette entrée passe donc de 1 à 0, et la 3e bascule ne bascule pas et le reste de la chaîne demeure en l'état.

Je ne vais pas développer l'explication plus loin. Vous n'aurez aucune peine à suivre les événements sur le tableau de la figure lorsque défilent les tops ultérieurs. Je vous conseille de le faire en détail, avec votre crayon, car vous éprouverez une

véritable joie à constater comment cet arrangement simple et harmonieux aboutit à faire apparaître sur les sorties utiles tout justement ce que nous voulions, à savoir la suite des mots binaires représentant les nombres 1, 2, 3, 4, 5, 6, 7, 8, ..., lorsque se succèdent 1, 2, 3, 4, 5, 6, 7, 8, ..., tops. Et, chose plus remarquable encore, le défilement des nombres, en binaire, reproduit exactement celui qu'illustre le tableau de la page 130, où la numération binaire a été exposée : le digit le plus bas, celui de rang 0 situé le plus à droite, bascule à chaque coup, le digit suivant de rang 1 bascule tous les deux coups, le digit suivant de rang 2 bascule tous les trois coups, etc. Si vous faites le petit travail que je vous recommande, vous sentirez que cette propagation de plus en plus lente des basculements quand on va vers la gauche, c'est-à-dire vers les rangs élevés, provient de ce qu'on ne parvient à un passage « aller » sur une bascule de rang n que lorsque toutes les combinaisons de passages « aller » et « retour » ont été épuisées sur toutes les bascules de tous les rangs inférieurs à n, ce qui a fait défiler tous les nombres inférieurs à 2^n.

Nous avons bien réalisé le compteur de nos rêves : il affiche le nombre des tops qui ont paru sur l'entrée depuis un début — marqué par le dernier RAZ — et le maintient affiché aussi longtemps que n'apparaît pas un top nouveau. Si on veut compter jusqu'à un total N préétabli, il suffit d'admettre la sortie du compteur sur l'une des deux entrées d'un comparateur de coïncidence dont l'autre entrée reçoit le nombre N : lorsque le comptage des tops atteint N, un signal émerge de ce comparateur et va alimenter la ligne RAZ, qui ramène le compteur à 0.

Le compteur est un organe de mémorisation très particulier, mais très utile et très élégant. Il termine ce chapitre en beauté.

Vous avez appris comment est bâtie et comment fonctionne une *mémoire*. On vous a présenté d'abord la brique fondamentale, une très fine perle trouée appelée *« tore de ferrite »*; et on vous a expliqué pourquoi son magnétisme peut présenter deux états distincts qui mémorisent un « top » ou un « tip », et entre lesquels le tore peut *basculer* en une fraction de microseconde. De là vous êtes passé à l'arrangement d'un temple où des colonnes de tores sont disposées sur un plan de base aux croisements d'une série de lignes x et d'une série de lignes y. Chaque colonne est une case de mémoire contenant un nombre N de cellules identiques égal au nombre des places d'un mot, et les deux coordonnées x et y de son pied forment son *adresse*. Vous avez vu avec admiration comment, lorsqu'on vise une adresse (x, y) en envoyant un signal sur la ligne x et un signal sur la ligne y, on n'impressionne que la seule case située à leur croisement, ce qui permet, soit de *« lire »* le mot qu'elle contient, soit d'y *inscrire* un mot nouveau. Vous avez constaté ensuite que les petits tores se prêtent à bien d'autres applications, en particulier à la construction de *chaînes à décalage* qui font émerger une impulsion d'une sortie après l'autre à mesure qu'on applique des tops à l'entrée. Après quoi, la passion des mémoires nous ayant saisis, nous avons vu comment on peut en fabriquer sans ferrites, avec des circuits logiques à transistors, circuits curieux appelés *« flips-flops »* ou *« bascules »*. Ces bascules permettent aussi de créer des *compteurs* capables de dire combien de « tops » on leur a appliqué depuis une dernière remise à zéro.

6

Puck

Ce chapitre sera très court. Jusqu'ici vous n'avez parcouru que des notions partielles et toisé que des éléments de construction séparés. Vous devez avoir hâte d'assembler toutes ces pièces dans un édifice fonctionnel. C'est ce que nous allons faire. Notre ambition ne consistera en rien moins qu'à bâtir un ordinateur complet. Un tout tout petit ordinateur, certes, et auquel nous ne demanderons qu'une toute toute petite tâche, mais qu'il accomplira du début à la fin sans sauter aucune étape. C'est pourquoi je vous propose de le baptiser du nom que porte le lutin agile et serviable de Shakespeare : Puck. C'est un modèle miniature conçu pour incarner les idées, et non pas pour travailler réellement. En suivant les cheminements de l'information dans le corps électronique de Puck, vous comprendrez à fond comment travaillent les ordinateurs véritables.

L'anatomie de Puck est dessinée sur la planche encartée à la fin du livre, et que je vous prie de déplier. Ne vous laissez pas affoler par ce schéma. Vous verrez qu'avec la science que vous avez acquise, et un peu ma compagnie, vous le démystifierez sans peine. Et consolez-vous en vous disant que le schéma d'un vrai calculateur nécessite des centaines de feuilles de plusieurs mètres carrés, couvertes chacune de milliers de traits, et assorties de 200 kg de liasses de nomenclatures, plans de câblage, spécifications, procédures de contrôle, etc. Ce schéma est simplifié aussi par la convention qu'un seul fil convoie une information. En réalité celle-ci est constituée par un mot ayant pas mal de digits, qui circulent tous à la fois en parallèle sur autant de fils distincts. Il faut donc prendre chaque ligne du schéma pour la figuration symbolique de tout un faisceau de fils.

Pour faciliter et colorer le langage, j'appellerai les circuits « ET » des « portes » et les circuits « OU » des « mélangeurs ».

Nous allons voir comment s'exécute un morceau typique de programme, morceau qui ne comprendra que trois ordres GAR, ADD et RAN. Comme ils forment une séquence, je les ai placés dans des casiers contigus de la mémoire, comme suit :

Adresse de l'ordre	Écriture de l'ordre	Effet de l'ordre
n	GAR p	transfère le contenu de l'adresse p de la mémoire vers un registre spécialisé R ;
$n + 1$	ADD q	additionne le contenu de l'adresse q de la mémoire au contenu du registre R ;
$n + 2$	RAN r	range le contenu du registre accumulateur dans l'adresse r de la mémoire.

L'organe désigné par « *aiguilleur* » dans le grand schéma est un registre qui s'emplit de l'information lue dans la mémoire et présente celle-ci, par les trois fils tracés en gras, à trois portes A, B, C. Ce transfert a lieu dès que l'aiguilleur reçoit un top du compteur. Nous allons voir quand et comment ces trois portes fondamentales A, B, C seront ouvertes ou fermées.

Au départ, nous supposons le compteur chargé avec le nombre *n*, adresse où le programmeur a logé l'ordre initial de notre petite séquence. Vous réalisez tout de suite que le compteur va jouer le rôle de cet adjudant dont je vous ai parlé au chapitre 1, qui récite, à mesure des tops qu'il reçoit, cette suite des adresses où nichent les ordres et que j'ai appelée le « vade-mecum ».

Quand le compteur débite le nombre *n*, adresse où il faut aller voir pour apprendre quoi faire, ce nombre *n*, divisé comme toute adresse en deux moitiés *x* et *y*, s'en va à travers les mélangeurs J_x et J_y vers les matrices de sélection d'adresse qui, comme vous savez, réagissent en aiguillant des signaux adéquats vers la case de mémoire ayant les adresses (*x*, *y*), c'est-à-dire la case N° *n*. Cette case débite son contenu sur le circuit de lecture, qui le fait entrer dans l'aiguilleur. Le compteur ayant envoyé un top à ce dernier en même temps qu'il livrait l'adresse *n*, l'aiguilleur présente le mot qui logeait à l'adresse *n* aux trois portes A, B, C.

Laquelle des trois allons-nous ouvrir ? Je sais que le contenu de la case *n* récité par le vade-mecum est un *mot* ***logique***, qui doit m'instruire de l'action à entreprendre. Je vais donc l'envoyer vers l'*organe de contrôle*, situé à gauche, et c'est donc la porte A que je vais ouvrir. Et je sais d'avance que par la suite, lorsque le mot émergeant de la mémoire sera *numérique* et non plus *logique*, c'est au contraire vers l'*organe de calcul* que je devrai l'aiguiller à travers l'une ou l'autre des deux portes B et C qui y mènent. Bref, il faut que la porte A soit ouverte quand une adresse parvient du compteur récitant le vade-mecum, et fermée quand une adresse parviendra d'ailleurs.

C'est pour réaliser cette condition qu'on emploie la circuiterie constituée par la porte H, le mélangeur F et le « NI » G. La porte H donne du signal sur la seconde entrée de la porte A, et l'ouvre donc, lorsqu'elle-même est ouverte, c'est-à-dire lorsqu'il y a du signal à la fois sur ses deux entrées, c'est-à-dire à la fois à la sortie du mélangeur F et du « NI » G. Or il y a du signal à la sortie de F quand une adresse sort du compteur, et il y a du signal à la sortie de G quand rien n'y entre, c'est-à-dire qu'aucune adresse ne sort d'ailleurs.

De quel « ailleurs » autre que le compteur pourraient aussi parvenir des adresses ? Évidemment du mot logique issu de l'adresse de démarrage qu'avait visé le vade-mecum. Ce mot logique en effet contient, comme vous savez, dans sa partie gauche un *ordre*, et dans sa partie droite une *autre adresse*, où siège cette fois le nombre sur lequel il faut exécuter l'ordre, c'est-à-dire l'*opérande*. Or le mot logique qui siégeait à l'adresse *n* vient de sortir par la porte A, d'après le jeu que nous venons de décrire. Ce mot logique sera donc partitionné en deux : d'une part l'ordre, en l'espèce « GAR », qui sera déposé dans le « *registre d'ordres* » R_o ; et d'autre part l'adresse *p* de l'opérande, qui sera déposée dans le « *registre d'adresses* » R_a. C'est lui qui constitue la seconde source possible d'adresses. On aura soin de vider les deux registres d'ordres R_o et d'opérandes R_a par des signaux RAZ (remise à zéro) qu'on peut emprunter au signal sortant du compteur par des fils dérivés de l'entrée de la porte H.

Le « *registre d'adresses* » R_a débite donc l'adresse *p* d'un opérande, comme toujours subdivisée en deux moitiés *x* et *y*, vers les matrices de sélection qui alimentent

la mémoire, à travers les mêmes mélangeurs J_x, J_y par lesquels passent les adresses de vade-mecum issues du compteur. Mais, sur le chemin, on a placé deux portes I_x et I_y dont les moitiés x et y de l'adresse issue de R_a n'alimentent chacune qu'une entrée, et dont il faudra que les autres entrées soient également excitées pour que l'adresse passe. Pourquoi avoir dressé cet obstacle? Parce que, à l'encontre des adresses de vade-mecum qui doivent *toujours* être visitées dans la mémoire à peine sorties du compteur, celles des opérandes qu'on dépose dans R_a ne doivent pas toujours être exploitées sur-le-champ. Nous allons rencontrer tout à l'heure des cas où il faut se ménager, entre l'apparition d'une adresse dans R_a et l'excitation de la mémoire à cette adresse, le temps de faire quelque autre opération ou préparatif. A cet effet on utilise le fait qu'un registre affiche toute information inscrite en lui, et la laisse lisible sans destruction, aussi longtemps qu'on ne le remet pas expressément à zéro. En conséquence, on interpose sur le chemin des adresses issues de R_a deux portes I_x, I_y, dont chacune reçoit sur l'une de ses entrées une moitié de l'adresse aussi longtemps que celle-ci siège dans R_a, et dont l'autre entrée va ouvrir le passage quand elle recevra un signal expressément envoyé parce que le moment sera venu d'appliquer ces adresses à la mémoire.

Nous entrons maintenant dans la séquence de programme proprement dite. Le mot logique que nous avons sorti de l'adresse n se trouve, quant à l'ordre, qui est « GAR », dans le registre R_o, et, quant à l'adresse d'opérande, qui est p, dans le registre R_a. L'heure de vérité a sonné, il faut agir. Qu'est-ce qu'on fait? Une grande matrice d'aiguillage à diodes, suivies de chaînes à décalage, a été brochée de telle manière que, sous l'action du code top-tip représentant l'ordre « GAR » et appliqué à l'entrée, des fils bien déterminés vont recevoir et porter des tops à des instants bien déterminés. J'ai appelé ce dispositif l' « ***interpréteur*** ». Les bornes de sortie de l'interpréteur et les fils qui sont actionnés par l'ordre « GAR » sont dessinés sur la figure en **rouge.** L'arrangement des diodes et des décalages est tel qu'au premier temps, sur le fil qui sort de la borne T_1, on recueille un top qui va remettre à zéro le registre R afin qu'il soit prêt pour accommoder le nombre issu de l'adresse p. Au deuxième temps on sort de la borne T_2 un top qu'on véhicule d'une part vers la porte de sortie B de l'aiguilleur, d'autre part vers les portes de sortie I_x, I_y du registre R_a. Il en résulte que le mot désignant l'adresse p, mot contenu dans R_a, peut maintenant passer vers la mémoire. Il y provoque la lecture du nombre (l'opérande) contenu à cette adresse p. Ce mot tombe dans l'aiguilleur et s'applique, comme tout ce qui réside dans l'aiguilleur, aux trois portes A, B et C. Mais la porte A est fermée par le jeu des organes F, G, H, que j'ai décrit au début, car cette fois nous avons une adresse qui émerge de R_a et non plus du compteur. La porte C est également fermée, car sa seconde entrée n'est pas, à ce moment, excitée. Seule la porte B est ouverte puisque le fil **rouge** parti de T_2 y a amené un top d'entrée. Elle laisse donc passer l'opérande qui se loge ainsi, comme on le voulait, dans le registre R préalablement vidé.

Le rôle de l'ordre **rouge** « GAR » semble ainsi achevé. Cependant il reste une besogne de ménage à faire : il faut vider les registres R_o et R_a afin qu'ils soient prêts à absorber d'autres informations, et actionner le compteur d'un cran afin qu'il passe au numéro suivant du vade-mecum.

Ce numéro est l'adresse $n + 1$ où siège l'indication de ce qu'il faut faire au prochain pas. Et le cycle précédent recommence. Le compteur délivre l'adresse

$n + 1$ à la mémoire tout en ouvrant la porte A. La mémoire débite le contenu de l'adresse $n + 1$, via l'aiguilleur, vers cette porte A. Ce contenu est un mot logique dont la partie « ordre » est l'instruction « ADD », qui va siéger dans le registre R_o; tandis que la partie « adresse », qui est q, va siéger dans le registre R_a. L'interpréteur a été broché de telle manière que, quand on lui applique le mot signifiant en code l'ordre « ADD », se déclenchent les événements marqués en **bleu** : au premier temps le fil issu de A_1 prépare, en le vidant, le registre accumulateur à recevoir la somme des deux nombres dont le premier, qui siégeait à l'adresse p, se trouve sur la porte D après l'exécution de l'ordre précédent, et dont le second, qui siège à l'adresse q, tombe sur la porte C parce que le registre R_a va appliquer cette adresse q à la mémoire. A cet effet, les portes de sortie I_x et I_y de R_a s'ouvrent par un signal débité depuis la borne A_2 à travers le mélangeur dans un second temps; lequel signal ouvre simultanément les portes C et D, en même temps que se ferme la porte A afin d'interdire cette voie aux opérandes. A ce moment donc, l'addition voulue s'exécute et le résultat s va siéger dans l'accumulateur. La séquence d'événements provoquée par le second ordre se termine, comme dans le cas du premier ordre, par un « top de remise à zéro » débité au troisième temps sur la borne A_3, top qui va remettre à zéro les registres R_o et R_a et pousser d'un cran le compteur.

Celui-ci débite maintenant le nombre $n + 2$, du vade-mecum, qui va, comme dans les deux pas de programme précédents, amener dans R_o et R_a le mot logique contenu à l'adresse $n + 2$. Ce mot contient l'ordre « RAN » et l'adresse de mémoire r où il faut ranger le contenu de l'accumulateur.

L'ordre « RAN », appliqué via R_o à l'interpréteur, provoque le déroulement des événements « **verts** » aux instants et aux endroits adéquats : au premier temps, un top venant de R_1, ouvre la porte E qui autorise le passage de la somme s contenue dans l'accumulateur vers les circuits d'inhibition, afin de préparer la mémoire à enregistrer le nombre s à l'adresse qu'on va indiquer; au deuxième temps, venant de R_2, un top passant par le mélangeur K autorise, en ouvrant les portes de sortie I_x et I_y de R_a, l'adresse r à passer sur la mémoire, et cela (comme j'espère que vous vous en souvenez) suffit à faire tomber le nombre s dans cette adresse, les inhibitions convenables ayant été préparées au premier temps. Enfin, au troisième temps, un top issu de R_3 va faire la besogne de ménage qui termine toute séquence d'opérations : la remise à 0 des registres et le petit coup d'avancement d'un cran sur le compteur.

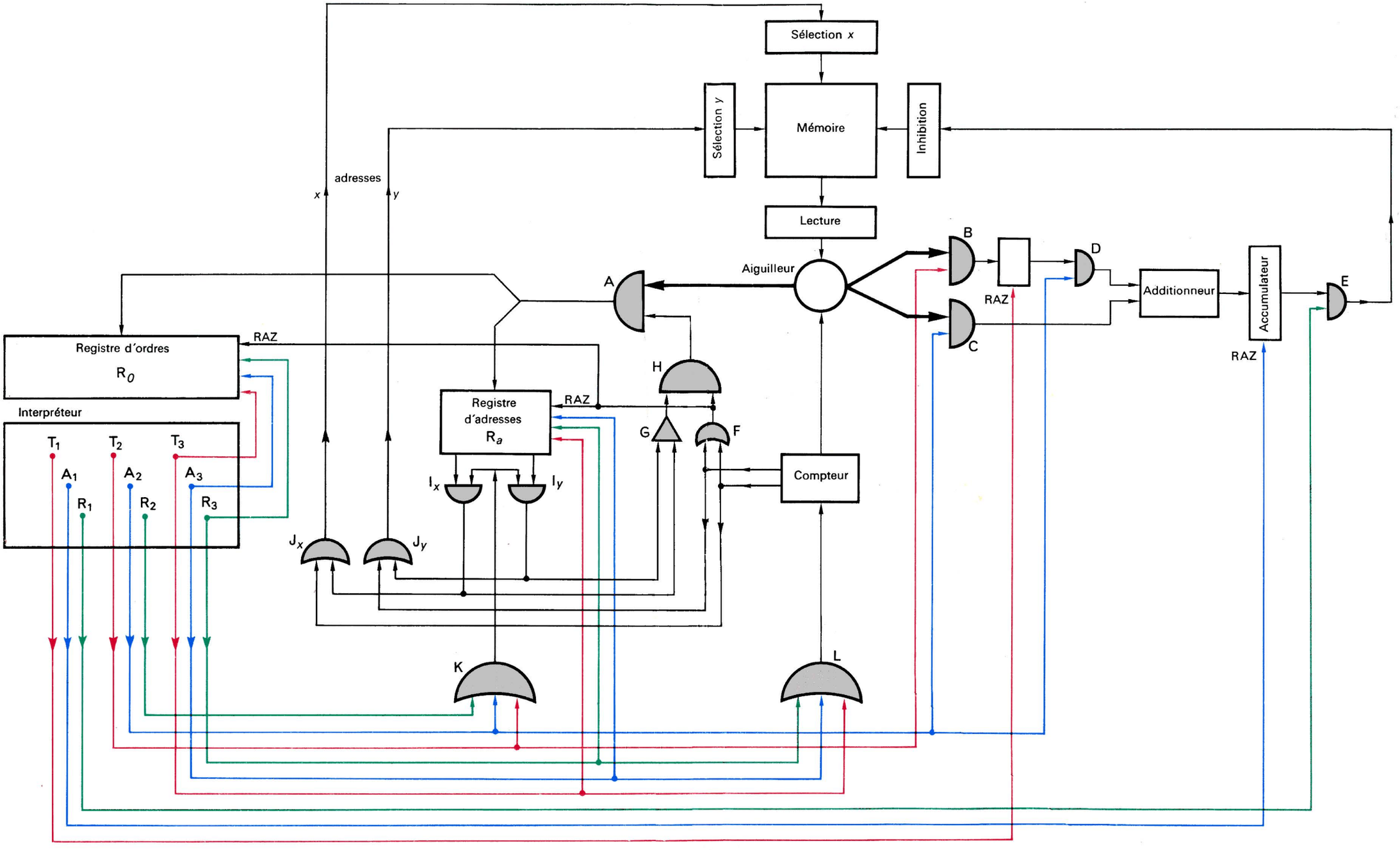
Et voilà. Le conte de Puck est terminé. J'espère que vous avez pleinement réalisé désormais ce que cela signifie, pour l'ordinateur, que de « comprendre » une instruction. Cela signifie qu'on a ***préparé l'interpréteur*** de façon que, lorsqu'on lui applique le mot représentant cette instruction en code binaire, il déclenche sur des fils adéquats et à des moments adéquats des tops qui vont ouvrir et fermer des portes adéquates, ou provoquer tout autre effet de commutation et/ou de mémorisation adéquat. Grâce à quoi l'information circule, se divise, se loge de manière que l'opération ordonnée s'accomplisse. Si vous vous rappelez que cette préparation de l'interpréteur consiste à le brocher de diodes et à lui accrocher des chaînes à décalage, disons en résumé à le ***harnacher*** d'une façon *irrévocable*, vous constaterez que nous n'avons conféré à l'ordinateur qu'une faculté très circonscrite. Il pourra « comprendre » d'*une certaine façon*, et *d'aucune autre*, *telles instructions précises*, et *aucune autre*. Nous avons certes créé le système nerveux d'un « cerveau », mais doté d'une intelligence excessivement spécialisée, capable seulement de réagir par des trains de secousses

internes exactement réglés d'avance à une liste exclusive d'ordres exactement définis d'avance. Je vous l'ai déjà dit, nous avons en somme créé la sorte d' « intelligence mécanique » que Descartes imaginait chez les animaux « sans âme ». Disons tout de même : des animaux parfaitement dressés. Et quant à l'absence d'âme chez notre esclave ordinateur, nous en reparlerons.

Dans le schéma de Puck vous reconnaissez clairement les trois grandes subdivisions que je vous ai déjà signalées chez tout ordinateur : — en haut, l' « *organe de mémoire* » avec ses auxiliaires; — à gauche et au centre : l'*organe qu'on appelle « de contrôle »* et qui comprend, outre une série de circuits de commutation, l'essentiel interpréteur où gît la véritable « intelligence » de la machine puisque c'est là que les ordres sont *reconnus* et que les signaux nécessaires à leur exécution sont engendrés; — enfin, à droite, ce qu'on appelle l' « *organe de calcul* », comprenant tous les circuits nécessaires à l'exécution effective des opérations. Mais, en vérité, il manque encore à Puck d'autres organes indispensables : ceux par lesquels il communique avec les hommes, ses maîtres et exploiteurs. Je ne vous ai en effet pas dit comment tous les ordres conçus par le programmeur sont introduits dans Puck et comment les résultats élaborés par Puck sont communiqués au dehors sous une forme facilement intelligible par l'homme. Vous savez que c'est là la tâche des organes qu'on appelle « *périphériques* ». Ils sont remarquablement astucieux et je vais vous les présenter dans le chapitre suivant.

Vous avez appliqué toute la science que vous avez acquise jusqu'ici pour construire vous-même l'ossature complète d'un mini-mini calculateur exécutant un mini-mini programme.

voir page 133 (chap. 6 : Puck).

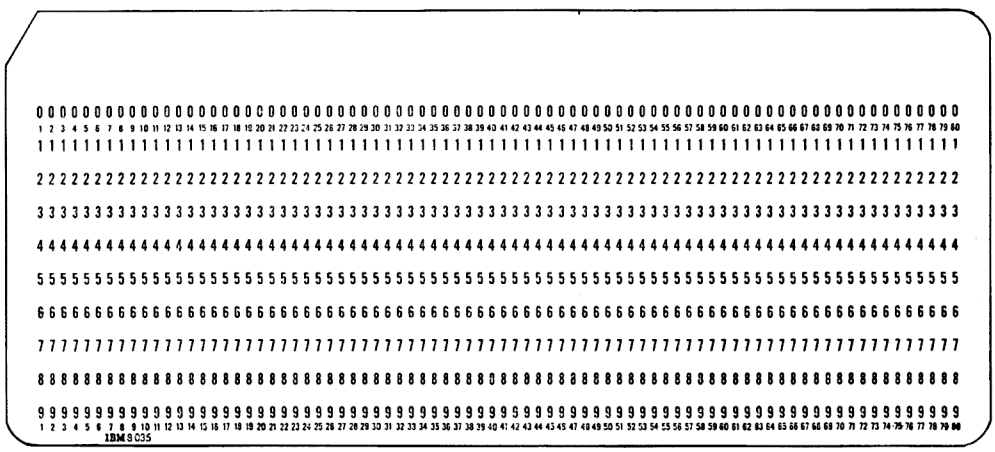


7

Les périphériques

Les cartes perforées

Le plus célèbre, le plus simple, et non le moins efficace des moyens de communication homme-machine est la « carte perforée » dont la figure ci-dessous donne une reproduction légèrement réduite.



Tout le monde a vu cet objet désormais aussi familier, aussi intégré dans notre civilisation, que l automobile. L'industrie en fabrique plusieurs centaines de millions d'exemplaires par an. Le modèle est standardisé au cinquantième de millimètre près. Le principe est exactement celui par lequel j'ai commencé ce voyage dans l'univers *ordino* : on inscrit une information, sous forme codée, en perçant ou ne perçant pas des trous standards à des endroits d'un carton préparés à cet effet et exactement repérés. Dans la carte standard les endroits préparés sont au nombre de 880, répartis en 11 lignes et 80 colonnes (il existe une douzième ligne, mais réservée à des tâches de contrôle et de ménage). Dans les premiers temps de leur emploi — et les inscriptions toujours imprimées sur ce carton portent encore témoignage de cette époque — on consacrait par exemple *une* colonne à inscrire *un* digit décimal, dont la valeur était donnée par la ligne où on faisait la perforation. On utilise aujourd'hui de nombreux

autres types de codage, et l'habitude est restée de signaler par un trou dans la onzième ligne que la colonne au-dessous n'est *pas* perforée à l'ancienne manière. La *colonne* reste tout de même le cadre de base de l'information inscrite dans la carte, ce qui crée une petite difficulté parce que normalement on *lit* les cartes *ligne par ligne*. Cela oblige en effet d'employer une petite *mémoire tampon* pour conserver les digits lus en ligne jusqu'à ce qu'on puisse les regrouper selon les colonnes d'où ils émanaient.

La perforation des cartes se fait dans des machines spéciales, car elle doit être de haute précision. Vous ne vous étonnerez pas d'apprendre que ces machines s'appellent des « *perforatrices* ». Ce sont des supermachines à écrire, qui guident la carte et la déplacent automatiquement devant la touche qui les frappe, laquelle touche porte un couteau aiguisé pour effectuer la découpe. Une dactylo spécialisée frappe sur un clavier les nombres et les mots préparés par un programmeur et écrits en clair, et la machine s'arrange pour les transformer en trous selon un code convenu. Dans des versions plus simples de la machine, le préparateur met préalablement au point un « plan » de perforation que la dactylo suit des yeux et de la main. La plupart des perforatrices modernes comportent aussi une mémoire tampon à l'entrée, dans laquelle on peut stocker l'information. Celle-ci alors se déroule toute seule et quand on veut, provoquant *automatiquement* les perforations convenables. Ces machines peuvent perforer jusqu'à 200 cartes complètes par minute.

Pour *lire* l'information qui fut enregistrée dans une carte sous la forme « trou ou pas trou », on emploie aussi, quant au fond, la méthode esquissée page 18 c'est-à-dire qu'on vient présenter la carte de 80 colonnes, ligne après ligne, devant une rangée de 80 rayons lumineux aboutissant à 80 cellules photoélectriques. Ces dernières donnent du signal — un top — ou pas de signal — un tip — selon que la place de la carte qui se trouve devant elles a été perforée ou non. Et ce signal entre dans l'ordinateur, soit pour être réinscrit dans une autre sortie de mémoire, soit pour être exploité séance tenante. On est bien passé de l'univers *homo* à l'univers *ordino*. Dans certains cas, au lieu de lire la carte par l'arrêt ou le non-arrêt des rayons lumineux, on la lit en la faisant glisser entre une rangée de 80 môles métalliques et une rangée de 80 pointes en regard appelées « balais », ces pointes étant légèrement poussées vers les môles. A l'endroit où la carte présente un trou, le balai vient contacter le môle, et un circuit électrique se ferme, permettant le passage d'un 1. Aux endroits non perforés, le circuit reste coupé par le carton, et il n'y passe rien, c'est-à-dire qu'il y passe un 0.

Mais ce n'est là que le principe. Pour le mettre en œuvre pratiquement, il a fallu développer des dispositifs électromécaniques très savants. Le problème en effet n'est pas mince de faire circuler très vite des cartes de 0,2 mm d'épaisseur de façon qu'elles viennent s'immobiliser devant des points lumineux ou des balais sans que le jeu dépasse un cinquantième de millimètre, de les faire descendre 12 fois de suite exactement d'un interligne à chaque coup (en faisant savoir à l'ordinateur sur quelle ligne elles se trouvent), puis de les remettre dans le flot.

Et on ne s'est pas contenté de ce travail pourtant déjà délicat. Un lot de machines électromécaniques ont été développées qui exécutent sur les cartes toutes sortes de tâches plus complexes. Les **« trieuses »**, après avoir laissé la mémoire tampon regrouper par colonnes les informations lues dans une première passe, répartissent les cartes, au cours d'une seconde passe, en 12 catégories : 1, 2, 3, ..., 10, 11, 12, selon le contenu de chaque colonne. On voit alors les cartes tomber d'un très joli mouvement

légèrement déhanché dans 12 réservoirs aux parois de verre. Et ce à des vitesses qui sidèrent chaque fois qu'on assiste à cette danse, puisqu'on arrive à barater ainsi, suivant une logique, jusqu'à 1 000 cartes par minute! Une « **interclasseuse** » (en anglais « **collator** ») peut faire sur les cartes des tris encore plus subtils, comme par exemple fondre en un seul deux paquets de cartes classées selon deux critères différents de manière à retrouver ensemble toutes les cartes qui, dans un paquet ou l'autre, répondent à un même sous-critère; ou encore extraire au vol d'un paquet, et ranger à part, toutes les cartes qui contiennent un code donné. Cette dernière opération utilise le « circuit de coïncidence » que je vous ai décrit page 94. Le code donné est mis dans un registre qui l'applique sur un côté de ce circuit de coïncidence, tandis que les tops issus de la lecture des cartes sont appliqués sur l'autre côté. Chaque fois que le circuit constate que les deux mots sont identiques, il émet un signal qui ouvre, sur le chemin de la carte reconnue conforme au code, un aiguillage vers un réservoir spécial. Les cartes non conformes s'évacuent dans une autre trappe. Les « interclasseurs » peuvent également traiter 1 000 cartes à la minute.

Il y a encore d'autres machines pour manipuler les cartes perforées, en particulier une « *reproductrice* » pour créer des cartes secondes identiques à celles d'un paquet modèle, et une « *tabulatrice* » qui non seulement lit des cartes pour faire entrer leur information dans l'ordinateur, mais encore procède à toutes sortes de comptages des tops engendrés et s'arrête ou repart selon que certains totaux sont atteints ou pas. En outre la tabulatrice peut imprimer en clair l'information qu'elle est en train de lire, selon un principe que nous reverrons.

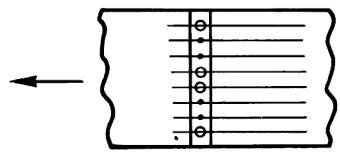
Je ne vous décrirai pas toutes ces machines, bien qu'elles jouent un rôle important en informatique, parce qu'elles ne posent pas de problème pour être comprises. C'est de la mécanique normale, encore que très astucieuse. Le plus remarquable est encore le carton lui-même, si parfait en planéité, souplesse, résistance et précision, qu'on peut conserver des cartes pendant des années en les faisant travailler tous les jours. Sous des formes rudimentaires, ces machines ont *précédé* les ordinateurs de plusieurs décennies. La carte perforée fut en effet inventée vers 1890 par l'Américain Hollerith pour aider le bureau de statistique de son pays à dépouiller les recensements de la population. Hollerith lui-même s'était inspiré d'un modèle illustre qui joua un rôle économique et social de premier plan, la fameuse carte où Jacquard, à l'orée du XIX^e^ siècle, inscrivit, également par des perforations, le plan des mouvements que devaient faire les broches pour exécuter un tissage donné. Jacquard faisait tâter ses perforations, qui déjà contenaient des formes embryonnaires de code, par des doigts mécaniques dont les faibles rebonds au passage des trous ou pas trous allaient commander les rouages du métier à tisser.

Les cartes perforées présentent toutes sortes d'agréments. Elles sont faciles à trouer, on peut les « lire » et vérifier avec ses yeux, on peut les grouper et conserver commodément, on peut écrire des notes dessus, on peut, grâce aux machines que je viens de citer, les classer de nombreuses manières utiles avant de transférer leur contenu dans l'ordinateur, on peut les remplacer isolément dans le sein d'un ensemble, etc. Et IBM vient de sortir un nouveau modèle qui tout ensemble est plus de moitié plus petit et contient davantage d'informations (96 colonnes au lieu de 80). C'est en raison de tous ces avantages que les cartes perforées continuent d'avoir la faveur des informaticiens. Mais, bien sûr, elles ne sont pas électroniques. Même quand mille cartes défilent en une minute, le flux d'informations qu'elles engendrent reste des milliers

de fois plus lent que celui qui circule à l'intérieur de l'ordinateur. C'est pourquoi le mode d'emploi normal consiste, si un programme et des données ont été perforées dans un paquet de cartes, à le transférer d'abord dans la mémoire de l'ordinateur puis à faire les calculs à partir de là. Le temps passé à ce préparatif est le prix qu'on paye pour l'énorme discordance de vitesse qui existe entre les deux types de traitements électronique et mécanique (ou électromécanique).

Les bandes perforées

C'est la plus fruste et la moins chère des « interfaces » homme-machine. Une bande perforée est tout simplement un long ruban de papier — d'un bon papier certes, mais simplement papier — ayant une largeur standardisée (de l'ordre de 2 cm).



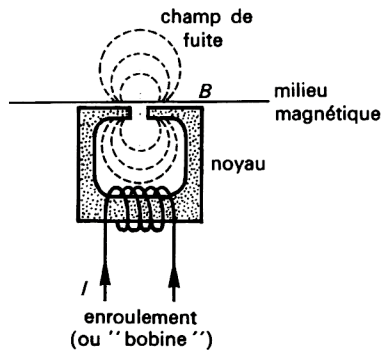
La bande est divisée, idéalement, en une succession de tranches jointives, perpendiculaires au sens du défilement. Sur chaque tranche, on dispose de 8 places exactement repérées, où on perfore ou ne perfore pas des trous. Pour ce faire, une tranche passe tout entière sous 8 couteaux alignés en travers de la bande. Quand ces couteaux s'enfoncent ou ne s'enfoncent pas sous l'influence d'un digit 1 ou d'un digit 0, ils découpent ou ne découpent pas des trous. Puis la bande présente aux couteaux la tranche suivante. Une tranche va ainsi accommoder un octet, ou petit mot de 8 digits qu'on appelle, dans le jargon spécial des bandes perforées, un « caractère ». Les perforatrices modernes permettent de découper 100 caractères par seconde. La lecture se fait de la même manière qu'avec les cartes perforées, mais plus vite puisqu'on peut lire aujourd'hui 1 000 caractères, c'est-à-dire 1 000 tranches *par seconde*, au lieu de 1 000 cartes *par minute*.

Les bandes perforées étaient déjà employées avant l'ordinateur pour y imprimer en code des messages *télégraphiques*, de sorte que les informaticiens ont pu puiser dans les équipements très efficaces développés par les télégraphistes. On a rassemblé aujourd'hui en une machine unique appelée « **Flexowriter** », toutes les fonctions d'écriture et lecture. On connaissait aussi les bandes perforées sous forme d'accordéons à grande largeur dans les pianos mécaniques de nos grand-mères. Les trous qu'on y pratiquait, en défilant, provoquaient l'enfoncement des touches dans un ordre voulu. Les joueurs d'orgue de Barbarie dans les rues utilisaient le même principe; ils entraînaient la bande à la main, par une manivelle, et cette silhouette étrange et nostalgique a servi dans bien des films.

Le gros désavantage des bandes est leur fragilité, et surtout qu'on ne peut changer un caractère sans changer toute la bande. Les bandes perforées ne sont plus guère utilisées que dans des installations modestes, ou lorsqu'on peut les traiter directement à la sortie d'une ligne télégraphique.

Le ruban magnétique

Avec les cartes perforées le temps d'accès et la capacité de stockage restent tout de même limités. Alors que la fringale des usagers de l'informatique n'a pas de bornes. Quand on veut par exemple faire élaborer par un ordinateur les ordres pour acheter du stock, pour aménager les paquets sur les rayons, pour facturer les ventes et régler les circuits de livraison dans une firme qui traite tous les jours mille tonnes de marchandises réparties en dix mille articles et huit cents succursales, vous imaginez aisément qu'il faut communiquer à l'ordinateur, et en extraire, tellement de données et des programmes si vastes, qu'en cartes perforées on remplirait trois salles au guichet desquelles on attendrait deux heures pour attraper une donnée ou un ordre particuliers. Il a donc fallu inventer des supports d'information stockée à capacité plus grande et à temps d'accès plus court.



Le plus répandu de ces supports est la *bande magnétique*. Elle exploite, mais avec des performances de plusieurs ordres de grandeur supérieures, le même phénomène que votre magnétophone familier. La bande est constituée par une très fine couche de matériau magnétique déposée au pistolet et au four, sur un substrat remarquable, un plastique appelé « mylar », ferme, fin et souple. L'épaisseur ne dépasse pas 1/10 de millimètre et la bande est normalement enroulée sur des bobines et se déroule devant ce qu'on appelle une « tête » magnétique. C'est un tout petit noyau (pas nécessairement rond) en fer doux, autour duquel est enroulé un fil conducteur et qui est fendu en un certain endroit devant lequel défile le ruban magnétique. Si on applique du courant électrique à l'enroulement, le noyau s'aimante et, à l'endroit de la fente, le champ magnétique qui le parcourt est obligé de se fermer *hors* du noyau par des lignes foisonnantes dites « de fuite ». Ce champ de fuite à son tour permet d'aimanter le point de la bande qui est en regard de la fente et qu'on suppose initialement vierge. On peut bien parler d'un « point », car on fait des fentes qui n'ont pas plus de 1/15 de millimètre de large et qui créent une dispersion du champ de fuite telle que la zone aimantable du ruban ne dépasse pas 1/30 de millimètre! Dès que le courant cesse, le noyau redevient neutre puisqu'il n'a qu'un magnétisme « mou », et la tête est prête pour une autre excitation. Inversement, s'il n'y a pas de courant dans la petite bobine, mais que le ruban est aimanté en une certaine zone, lorsque cette zone passe devant la tête le champ magnétique qui en émane et l'entoure, en

balayant la fente, induit du courant dans le bobinage; et ce courant, amplifié, témoigne du passage d'une zone aimantée. L'aimantation que prend le ruban est proportionnelle au courant qui l'induit, et, à l'inverse, le courant qu'on recueille au bobinage est proportionnel à l'aimantation du ruban qui passe.

On a ainsi un organe *bilatéral*, qui permet dans un sens d' « **inscrire** » des taches magnétiques sur le ruban en conformité avec le courant *inducteur*, et dans l'autre sens de « **lire** » d'après le courant *induit* les taches magnétiques qui ont été préalablement inscrites dans la bande. Vous observerez que le phénomène mis en jeu ici n'est pas du tout le même qu'avec les tores. Il n'y a aucun « basculement » de l'état magnétique d'un matériau « dur ». Il y a simplement *passage* du matériau devant une tête. En particulier, la lecture, par la tête, de l'information inscrite dans la bande sous forme de successions de taches magnétiques, *ne détruit pas celle-ci*. Si on veut l'effacer, il faut *écrire par dessus*, l'information nouvelle se substituant alors à l'ancienne comme le savent bien tous les usagers du magnétophone.

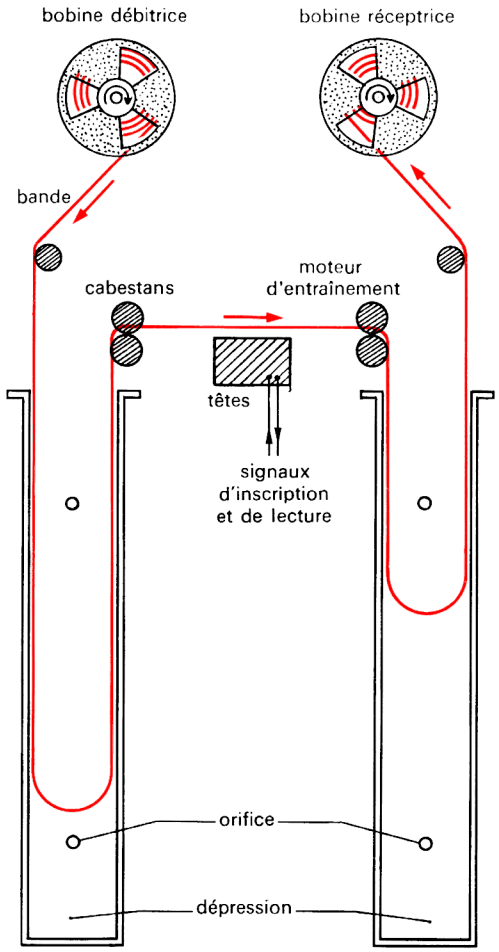
Dans le magnétophone ordinaire, destiné à enregistrer et livrer du texte parlé (ou de la musique), on emploie une bande étroite qui est impressionnée plus ou moins fortement, selon l'intensité du son à chaque instant. *Une seule* tête « module » ainsi l'aimantation locale de la bande en accord avec les évolutions du son dans le temps, et *une seule tête* lit cette aimantation pour restituer le son.

L'emploi est tout différent en *ordino*. Vous savez depuis le premier chapitre qu'il n'est pas question, en *ordino*, de s'en remettre à des fluctuations continues d'intensité. On n'aimante la bande que par « tout » ou « rien » au moyen de tops identiques d'amplitude standard, et c'est l'ordre de succession de ces « oui » et « non » qui caractérise le message. En fait le « oui » et le « non », ici, ne sont pas vraiment des taches aimantées ou pas aimantées, ce sont plutôt des taches aimantées dans un sens ou dans le sens opposé; mais vous n'avez pas besoin d'entrer dans cette finesse.

On va donc, en *ordino*, traiter le ruban magnétique exactement comme le ruban de papier, à ceci près qu'au lieu de trous on a maintenant des taches magnétiques qu'on peut inscrire ou effacer à volonté. On rend le ruban assez large — par exemple 2 cm entre bords — et dans une transversale on installe 6, 7 ou 8 têtes côte à côte. De ce fait l'information est déposée par petites tranches transversales de 6, 7 ou 8 digits qui sont écrits ou lus simultanément. On arrive à inscrire et lire jusqu'à 30 tranches discernables dans un segment de bande de 1 mm de longueur.

Un autre trait distingue l'usage de la bande magnétique en *ordino*. Les exigences de précision en *ordino* sont tellement plus fortes qu'en *homo* qu'on ne saurait compter sur le déroulement même de la bande pour présenter tel ou tel morceau d'information *selon le temps mis*. Comme les tranches ne sont distantes entre elles que de 3 centièmes de mm, il suffirait d'un jeu de cette infime valeur pour qu'on lise la mauvaise tranche. Or un tel jeu est inévitable, car si inscription et lecture sont électroniques, le déroulement de la bande, lui, relève de l'imparfaite et lourde mécanique.

Pour pallier ce danger, on a dû prendre deux mesures, l'une logique et l'autre mécanique, toutes deux ingénieuses. La mesure logique consiste à n'inscrire de l'information que par *blocs* d'un certain nombre de tranches, mettons 1 800 tranches qui occupent donc 6 centimètres de bande. Quelques tranches spéciales en tête de chaque bloc, ou une piste latérale spéciale, sont réservées pour porter le *numéro* du bloc dans la bande, qu'on peut aussi appeler son *adresse*. L'adresse d'un bloc est lue au



vol pendant que la bande défile, et c'est elle, et non le temps de défilement, qui est retenue comme indice qu'on se trouve bien dans le bloc souhaité. Chaque bloc est ainsi identifié correctement même si le déroulement n'est pas parfait.

Encore faut-il que l'information comprise *à l'intérieur* d'un bloc défile avec la plus grande régularité possible, afin que les digits et les mots tombent en synchronisme avec les tops d'horloge de l'ordinateur. A cet effet on isole le morceau de bande portant le bloc, avec des blancs généreux en amont et en aval, entre deux boucles de « mou », et on applique une traction *indépendante* à la partie isolée, traction qui peut maintenant être de très haute précision parce qu'elle n'intéresse qu'un segment petit et fixe de la bande *non chargé* par le poids complet des longues parties de bande qui se trouvent en amont et en aval.

On arrive ainsi au « dérouleur » devenu classique et représenté par la figure ci-dessus. Une bobine débitrice animée par un moteur indépendant livre une boucle de réserve dans le tunnel de gauche dont la partie inférieure, *sous* la bande, est liée à une petite pompe à vide qui y établit une dépression de l'autre côté. On laisse aussi

du « mou » dans le tunnel de droite d'où la bande sort pour être recueillie par une bobine réceptrice également entraînée par un moteur indépendant. La bobine débitrice débite jusqu'à ce que la boucle de gauche tombe *au-dessous* de l'orifice inférieur par où se fait la dépression. A ce moment cet orifice émerge à la pression atmosphérique qui règne *au-dessus* de la bande, et un relais pneumatique actionné par ce changement de pression s'en va stopper la bobine débitrice : on a accumulé assez de « mou » pour être sûr que le bloc qu'on veut exploiter est sur lui. Le moteur spécial d'entraînement de la bande se met alors en marche, et il peut la mouvoir avec une précision extrême car il n'a à tirer qu'un morceau de très faible poids. On s'est en somme *débrayé* de l'inertie qui charge l'amont. Le morceau de bande ainsi isolé passe devant les têtes magnétiques (il y en a autant côte à côte en travers de la bande que la tranche doit contenir de digits). Ce morceau de bande qui travaille s'en va, au-delà du moteur d'entraînement, former une seconde boucle de « mou » dans le tunnel de droite. Aussitôt que cette boucle passe *en dessous* de l'orifice supérieur situé dans ce tunnel, celle-ci, « basculant » en sens inverse du cas précédent, c'est-à-dire de la dépression à la pression atmosphérique, déclenche la bobine réceptrice qui se met à absorber la bande émergente.

Vous voyez que la bande ne prend ou ne livre de l'information que pendant les périodes intermittentes où elle passe sur les têtes. Ce fonctionnement par à-coups qui fut l'invention mécanique essentielle dans notre affaire, vous explique pourquoi les bandes au travail exécutent ces saccades — faisant penser à des toquades — qui ont dû vous frapper chaque fois que vous en avez vu une dans la vie ou en film.

Le dérouleur de bande magnétique est devenu lui aussi, comme la carte perforée, un objet familier. Une charge, c'est-à-dire une bobine, contient en général de l'ordre de 700 mètres de ruban qui défile en moyenne à la vitesse linéaire de 15 mètres par seconde. Si on enregistre avec une densité, aujourd'hui maximale, de 400 tranches par centimètre et compte tenu des espaces morts importants qu'il faut laisser entre les blocs, on constate qu'on peut emmagasiner sur une bande environ *30 millions de tranches d'information faisant 240 millions de digits utiles.* C'est évidemment beaucoup plus que n'en permettent les cartes perforées dans le même volume. Et combien de temps faut-il pour ***accéder*** à un bloc déterminé? Ici, attention, s'introduit une idée-clé : alors qu'une mémoire à tores *où rien ne bouge* permet de viser n'importe quelle adresse *en un seul temps machine*, la mémoire sur bande est comme on dit « **séquentielle** » c'est-à-dire que pour pouvoir lire ou écrire de l'information en un certain endroit, *il faut attendre que cet endroit ait défilé* devant les têtes. Comme la bande complète met environ 5 minutes à défiler, si on compte qu'en moyenne le bloc qui m'intéresse se trouvera à mi-chemin, il faudra donc attendre en moyenne 2 minutes pour y accéder. C'est énorme par rapport au temps d'accès à une adresse quelconque dans une mémoire immobile à tores, lequel vaut 1 à 2 microsecondes! Par contre, la mémoire statique arrive à grand peine à accommoder 1 million de mots; en général, elle ne dépasse pas 100 000 mots, alors que nous venons de voir qu'une bande peut en contenir normalement jusqu'à 30 millions.

Cette discordance phénoménale entre les deux types de mémoires est responsable de bien des casse-têtes chez les informaticiens. Passe encore si les bandes n'étaient utilisées *que* comme périphériques proprement dits, car pour *un* problème on n'a en général qu'*une* fois à entrer dans la machine et *une fois* à en sortir, et la rupture de rythme n'est alors pas trop grave. Mais il n'en est pas toujours ainsi. L'informaticien

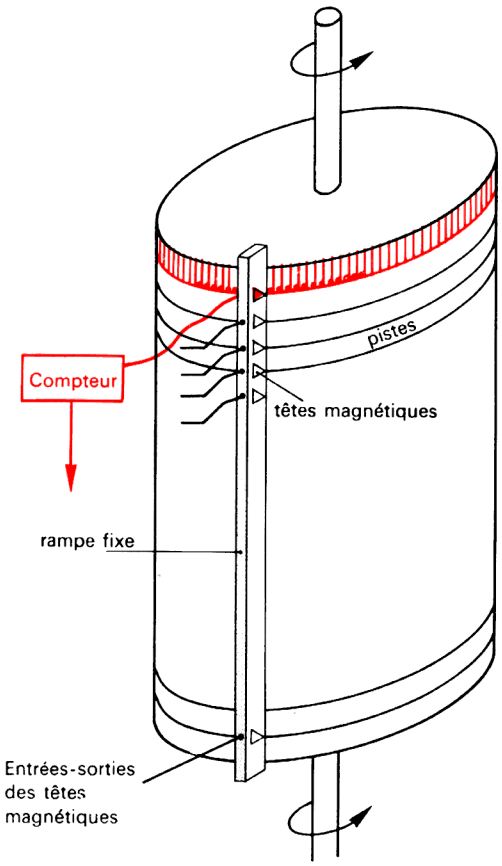
ne saurait résister à l'attrait que confère à la bande magnétique sa très grande capacité. Aussi s'en sert-il non seulement comme organe d'entrée et de sortie, mais encore comme *mémoire auxiliaire dans le travail même de l'unité centrale*. Dès lors il doit dépenser des trésors de soins pour ***organiser*** le meublement de la bande de façon que les blocs dont il aura besoin aux diverses phases du programme soient aussi *contigus* que possible. En règle générale, lorsqu'une bande sert ainsi de mémoire auxiliaire, on travaille par une succession d'allers et retours. Pendant, mettons, 2 millisecondes, on recopie trois ou quatre blocs contigus de la bande dans la mémoire statique de tores. Pendant les 2 millisecondes qui suivent, celle-ci travaille à son hyper-vitesse dans l'unité centrale, exécutant donc plusieurs milliers d'opérations, en même temps que la bande continue à défiler, présentant soit un blanc, soit des blocs d'information relevant d'un autre problème. Puis s'ouvre à nouveau un cycle comme le précédent.

Les bandes remplies de tops-tips magnétiques sont devenues des marchandises semblables à des gros livres. On les stocke dans des bibliothèques. On vend et achète des masses de données, ou des programmes très complexes, imprimés sur des bandes que la poste vous expédie dans des cartouches assez semblables aux bobines de cinéma. Signe que l'informatique est parvenue à maturité.

Les tambours magnétiques

Un tambour magnétique possède normalement une capacité inférieure à celle d'une bande, mais en revanche un temps d'accès beaucoup plus court. C'est un gros cylindre de 20 à 100 cm de diamètre, de 30 à 200 cm de haut (mais on construit aujourd'hui des tambours « super-géants » qui ont 2 mètres de diamètre et 5 mètres de hauteur). Il est couvert par une couche de poudre magnétique analogue à celle employée dans les bandes et tourne à raison de 50, voire 100 tours par seconde. La vitesse de 50 tours par seconde est préférée parce qu'elle concorde avec la fréquence du secteur électrique. Une rampe verticale, c'est-à-dire exactement parallèle à l'axe du cylindre, est installée à très très faible distance de la surface, souvent pas plus de quelques microns, si près que pour écarter le risque que d'éventuelles rugosités (infimes) de la poudre ne heurtent la rampe, il faut faire circuler autour d'elle un flux d'air sous pression. *Sur* la rampe fixe sont installées un certain nombre de têtes magnétiques équidistantes, par exemple tous les 2,5 millimètres. Sur une hauteur de, mettons, 60 cm, il y aura donc 240 têtes. Leur présence fait que la surface se décompose en 240 pistes circulaires, courant le long de parallèles du cylindre, et ayant chacune une longueur développée de, mettons, $\pi . 30$ cm $\approx$ 1 mètre si le diamètre du tambour est 30 cm. Le long de ces pistes on peut placer, mettons 20 taches magnétiques discernables par millimètre, soit 20 000 sur toute la piste, soit 48 millions de digits au total s'il y a 240 pistes, alors qu'une bande magnétique peut en contenir six fois autant. Seuls les tambours géants atteignent et dépassent la capacité d'une bande. Le plus gros du monde fait 2 milliards de digits. Mais, en revanche, le temps d'accès à un digit est considérablement plus court que sur une bande.

Combien vaut-il? C'est facile à voir. Une information est naturellement écrite au vol le long d'une génératrice du cylindre *lorsqu'elle passe devant la rampe*, et elle est lue de même. Si en moyenne l'information qui m'intéresse est située sur la

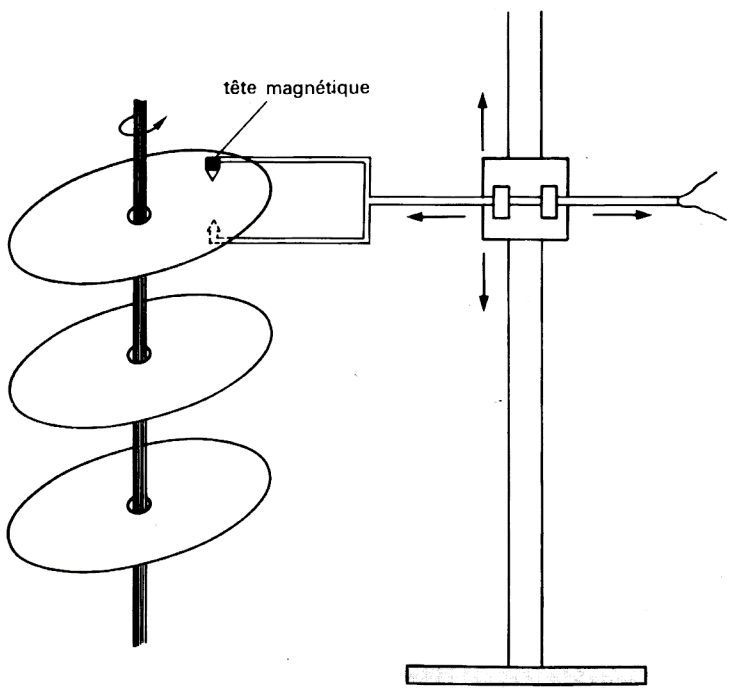


génératrice diamétralement opposée à la rampe, il faut attendre un demi-tour pour y accéder, soit 1/20 de seconde. Contre 2 minutes pour une bande! Le tambour est une mémoire *séquentielle*, lui aussi, mais *des centaines de fois plus rapide que la bande.* Son gros désavantage est son encombrement et sa lourdeur. C'est un éléphant qui ne peut travailler qu'à poste fixe, il n'est pas question de le faire circuler par la poste ni d'en stocker quelques-uns. (Un tambour géant exige un petit hall pour lui tout seul.) Le tambour magnétique sert surtout d'organe intermédiaire, sur lequel on copie une grosse masse d'informations qui devra travailler avec l'unité centrale sans trop pénaliser le rythme ultra-rapide de cette dernière.

Pour assigner une « adresse » à un digit on réserve tout en haut une piste sur laquelle on a inscrit une fois pour toutes et partout des 1. La tête qui lit cette piste est connectée à un compteur qui compte les 1 qui défilent, donc les génératrices, à partir d'une première jusqu'à la dernière, et recommence. En présentant à un circuit de coïncidence d'un côté le numéro de la génératrice sur laquelle on veut travailler, de l'autre côté le nombre qui sort du compteur, on peut *ouvrir* la tête qu'on veut au moment où les deux nombres coïncident, et ainsi viser toute cellule prédéterminée de la surface sensible. Très souvent on aménage plusieurs rampes sur le pourtour du cylindre, ce qui raccourcit encore le temps d'accès.

Les disques magnétiques

Il est un peu bête d'occuper tout le volume d'un gros tambour et de n'exploiter pour l'information que la seule surface externe. Les disques magnétiques écartent cette objection, car ils sont couverts de matériau magnétique sur leurs deux faces. Sur un disque de 40 ou 50 cm de diamètre on arrive à inscrire 200 à 300 pistes circulaires, soit mettons, 500 pistes par disque, sur chacune desquelles on peut tasser en moyenne 2 500 taches magnétiques à raison de 20 au millimètre. Un seul disque est donc capable de 500 × 2 500 soit 1 250 000 digits. On empile jusqu'à 50 disques sur un même axe, totalisant environ 60 millions de digits. Certains empilements vont jusqu'à 80 millions, c'est-à-dire 10 millions d'octets, autant ou plus que les plus grosses bandes magnétiques. Pour accéder à une piste, on fait avancer et reculer le bras porteur des deux têtes, et on est parvenu, grâce à des miracles de mécanique, à réaliser ces mouvements par des secousses si rapides que l'œil a peine à les suivre — 1/10 de seconde — et cependant si précis que la tête ne manque jamais la piste qu'on lui ordonne d'aller visiter, bien qu'elle ne soit distante de ses voisines que d'une fraction de millimètre. Ce bras ressemble un peu à la langue du lézard qui sort et rentre en un éclair sans jamais manquer sa proie. Pour chercher un disque parmi les autres on peut soit déplacer le bras le long d'une potence verticale placée à côté de la pile de disques, comme il est montré sur la figure, auquel cas la durée de ce déplacement s'ajoute à celle de la langue qui sort et porte le temps d'accès à 0,5, voire 0,8 seconde, soit disposer d'autant de bras qu'il y a de disques et exciter le bras voulu, auquel cas on évite le supplément de temps, mais on paye plus cher. Quant au temps de parcourir une piste, il ne compte pas dans le bilan car la pile des disques



tourne tout entière à 50 tours par seconde. Comme ils sont parfaitement dressés même à cette vitesse de rotation, ils ont l'air immobiles et on ne voit que les sautes d'humeur d'apparence fantasque de la langue de lézard.

Contrairement aux tambours, les disques sont amovibles, soit individuellement, soit en groupes entiers. C'est pourquoi leur emploi est de plus en plus répandu. On ne vend plus guère d'ordinateur qui n'ait quelque part son placard à disques.

En résumé, les trois grands types de périphériques magnétiques présentent les caractéristiques suivantes :

	Bandes	Disques	Tambour moyen	Tambour géant
Capacité	dizaines de mégabits	un peu moins	dix fois moins	dix fois plus
Temps d'accès	2 minutes	0,5 seconde	0,05 seconde	0,05 seconde
Souplesse de maniement	extrême	bonne	faible	nulle
Prix (d'une unité)	100 000 F	200 000 F	400 000 F	2 000 000 F

Comment lit-on les résultats?

Il ne suffit pas que l'ordinateur ait résolu un problème. Il faut encore qu'il nous fasse connaître la solution sous une forme qui convienne à nos habitudes d'hommes.

L'idée de l'idée à ce sujet est un peu la même que celle qui a conduit aux langages de programmation. Puisque des programmes suffisamment sophistiqués peuvent faire accomplir à la machine des tâches très savantes, comme de leur faire « traduire » dans le vocabulaire tops-tips utilisé par la machine des instructions écrites par le programmeur en clair, pourquoi ne pourrait-on pas faire l'inverse, c'est-à-dire créer un programme qui fasse traduire par la machine en textes clairs les résultats qu'elle a trouvé en tops-tips. Reprenons par exemple notre ordinateur du début qui assortit les couples désireux de se marier. A la demande de Monsieur *X*, notre ordinateur a trouvé que les postulantes suivantes conviennent :

- *1re variante* : Mlle *A* si M. *X* n'est pas trop difficile concernant la famille;
- *2e variante* : Mlle *B* avec un tour de poitrine supérieur au spécifié de 105 cm;
- *3e variante* : Mlle *C* satisfaisant à toutes les conditions sauf qu'elle a une licence d'anglais au lieu de son conservatoire de piano.

Dans la machine, les noms *A*, *B*, *C*, sont codés en tops-tips. De même la situation « famille douteuse » aura reçu le code α (α étant une certaine suite de tops et tips); la situation « tour de poitrine de tant » aura reçu le code β suivi du nombre *n* de centimètres, le tout en tops-tips; et les situations « licence d'anglais » et « conservatoire » auront reçu respectivement les codes γ et δ, le concept « au lieu de » étant

exprimé par le code ω. La réponse de la machine à la demande de M. *X* aura donc été élaborée dans le ventre de la machine sous la forme des expressions binaires suivantes, qui seront tombées dans une zone de la mémoire réservée aux résultats finaux des calculs :

1 : *A*, α
2 : *B*, β, *n*
3 : *C*, γ, ω, δ.

On peut se contenter de laisser ces réponses se déposer telles quelles dans cette zone de la mémoire, puis de lire celle-ci. Même dans ce mode d'exploitation naïf, se pose le problème de la forme pratique de cette « lecture ». Car jusqu'ici, quand nous parlions de « lire », le contenu de la mémoire, nous nous placions *dans le sein de l'univers ordino*, où la lecture se matérialise simplement par une suite de tops et tips. Il n'est pas impossible que l'exploitant de l'ordinateur s'en contente, par exemple, s'il destine le résultat à être réintroduit dans l'ordinateur pour de nouveaux calculs. Dans ce cas la solution est simple : les tops et tips représentant les trois réponses précédentes, et qui sont automatiquement débités par la mémoire quand on lit sa zone spéciale de résultats, sont dérivés, grâce à un signal *ad hoc*, vers des *bornes de sortie* qu'on connecte à une *perforatrice* de cartes ou de rubans. Cette machine assure alors un percement de trous représentant les trois messages précédents *en code*, et on retrouve ceux-ci en tops-tips sur le carton ou le papier, qu'on garde pour usage ultérieur.

▶▶ — Mais non! vous écriez-vous. Vous m'avez dit que l'inscription sur des cartes marche 10 000 fois plus lentement que la lecture d'une mémoire. Si je devais attendre pour consulter une nouvelle case de la zone « résultats » de la mémoire qu'un trou ait été perforé dans une carte, j'imposerais à la première une oisiveté insupportable.

▶ — C'est bien vrai. Aussi ne passe-t-on jamais *directement* l'information de la mémoire centrale de l'ordinateur vers des cartes. On dispose d'une *mémoire tampon*, en général installée dans le meuble des cartes, et on commence par y envoyer l'information à sortir, ce passage de la mémoire centrale vers la mémoire tampon s'effectuant à l'hypervitesse qui règne dans l'univers électronique. Ceci fait, la mémoire centrale peut vaquer là à d'autres besognes, et on prend son temps pour débiter la mémoire tampon vers la perforation de cartes. Cette mémoire tampon ne recevra pas d'autre tâche à faire avant qu'elle se soit vidée, le programmeur y aura veillé. Une découpe idéale de la chronologie des événements consiste à faire marcher une séquence de nombreux travaux internes à hypervitesse pendant que les résultats d'une séquence antérieure sont transcrits sur les cartes.

Mais ce système reste primitif en ce qu'il ne fournit qu'une copie *littérale* sur cartes, sous forme de trous et pas trous, du message en réponse 1 : *A*, α — 2 : *B*, β, *n* — 3 : *C*, γ, ω, δ tel qu'il a été élaboré sous forme de tops-tips dans l'ordinateur. Or cette forme du message reste rébarbative. Rien ne nous empêche de concevoir un programme qui instruise la machine, si elle reçoit le message précédent, d'engendrer les codes de toutes les lettres d'une série de mots que j'ai préparés d'avance pour exprimer le contenu du message d'une façon plus amène, par exemple, j'aurai décidé que le message : *A*, α s'exprimera par la phrase : « votre premier choix serait M^lle *A* (ici nom en toutes lettres) si vous n'êtes pas trop chatouilleux sur la famille dont le

standing est douteux ». Je peux même m'arranger pour que la machine élabore la suite des codes de toutes les lettres (et des blancs) de tous les mots qui forment la fiche descriptive de M^lle *A*. Ce programme qui effectue l'anamorphose du message binaire schématique en un texte « riche » et écrit en clair, s'appelle couramment le « *programme d'édition* ». Je le dépose dans la mémoire centrale de l'ordinateur et le fais exécuter par lui aussitôt après qu'il a établi la forme binaire du message raccourci répondant au problème posé. Le résultat du travail qu'appelle l'exécution de ce programme d'édition est donc la suite des codes binaires de la phrase qu'il faut sortir pour renseigner l'homme dans son registre discursif habituel. Ce texte (qui, attention, reste encore *codé signe par signe*) est également déposé dans une mémoire tampon. Il s'agit maintenant de transformer cette suite de codes de signes en lettres et signes typographiques écrits noir sur blanc sur du papier.

Le premier moyen dont on dispose pour cela est une *machine à écrire électrique*. Dans une telle machine les touches peuvent être actionnées non plus par la frappe manuelle de la dactylo, mais par un signal électrique qui s'applique à la touche derrière le capot. Or nous savons très bien, par la matrice à diodes, aiguiller un signal vers un endroit voulu en fonction du code appliqué, donc exciter la touche portant la lettre *A* si la mémoire tampon débite le code de la lettre *A*, la touche *B* si la mémoire tampon débite le code de la lettre *B*, etc. On voit ainsi les touches s'abaisser sans dactylo, comme si le clavier était frappé par l'homme invisible. Et sur le papier se forme devant nos yeux la phrase : « Votre premier choix serait M^lle *A* si vous n'êtes pas..., etc. » On arrive à faire s'imprimer ainsi au maximum 12 caractères par seconde.

Les imprimantes

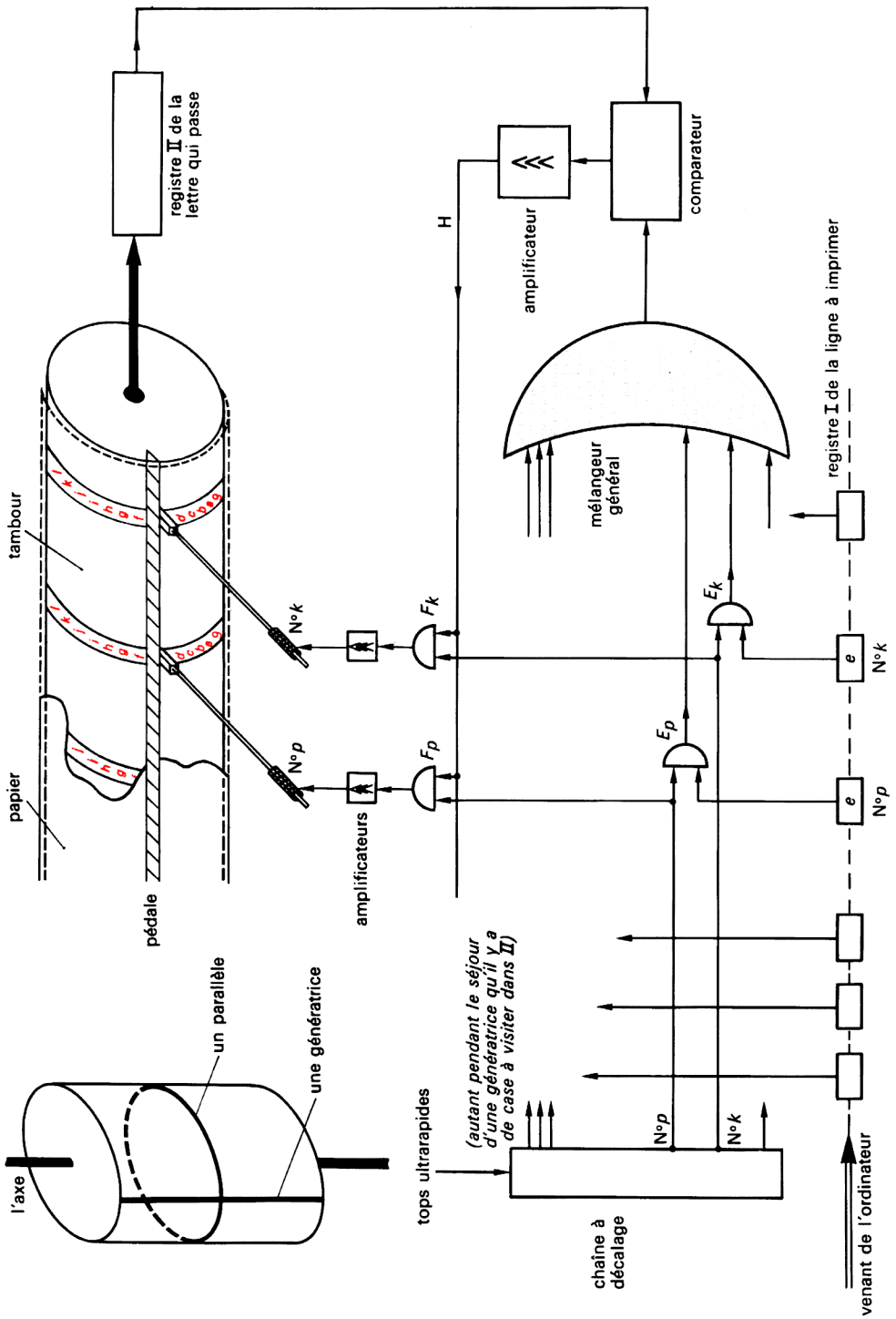
C'est quand même peu, fût-ce pour une machine électromécanique. La solution n'est acceptable que dans des cas simples et des installations modestes. On peut faire mieux, beaucoup mieux, grâce à un système que je situe pour ma part au sommet de l'ingéniosité et devant lequel, quand je le vois abattre sa besogne cyclopéenne dans le temps d'ouvrir et de fermer les yeux, je ne peux m'empêcher d'éprouver un frisson d'effroi sacré. Voyons cette forge.

Le *programme d'édition* a donc défini tous les codes de tous les signes qui doivent se succéder pour former le texte plus ou moins fleuri où l'homme lira la réponse à son problème trouvée par la machine. Le programme d'édition a même fixé la découpe de ce texte en lignes, de façon que chaque ligne comprenne, mettons, 80 caractères, un espace ou un signe typographique comptant pour *un* caractère. Considérons l'une de ces lignes. Sous l'action du programme d'édition, les 80 codes binaires des 80 caractères qui devront la composer ont été déposés dans les 80 cases du registre I qui occupe le bas de la figure. D'autre part, une usine (très raffinée) m'a construit un grand tambour sur lequel sont déposés, le long de 80 parallèles, 80 bandes identiques portant chacune la totalité des signes qui peuvent entrer dans l'écriture en clair des messages, mettons les 26 lettres de l'alphabet, les 9 chiffres et 5 signes typographiques, soit 40 caractères au total. Vous vous rappellerez que sur un cylindre on appelle « parallèles » les cercles courant dans des plans perpendiculaires à l'axe et « génératrices » les droites parallèles à l'axe (voyez le coin en haut à gauche de la figure, page 154).

Les 40 caractères d'une bande courant le long d'un parallèle sont sculptés en relief sur la surface avec une extrême finesse et précision, et encrés. Une même lettre se retrouve très exactement au même endroit sur les 80 bandes parallèles, de sorte qu'on peut parler de « la génératrice $\boxed{m}$ », « la génératrice $\boxed{7}$ », etc. du tambour. Parallèlement à l'axe du tambour, et à un ou deux dixièmes de millimètre de sa surface, on a dressé une longue pédale en métal ferme mais souple. Un papier passe dans l'interstice et est entraîné par un mécanisme que je ne représente pas. Enfin, derrière la pédale, se tiennent 80 petits marteaux, un en face de chacune des 80 bandes porteuses de caractères. Ces petits marteaux sont portés par des tiges qui coulissent dans des électroaimants. Si un électroaimant reçoit du courant, la tige qui le traverse subit une force qui fait frapper d'un coup rapide et sec le marteau correspondant sur la pédale, et cela appuie le papier contre la surface du tambour, avec pour résultat qu'à cet endroit apparaît la lettre qui se trouvait défiler à cet instant devant la pédale sous le marteau excité. Un dispositif facile à imaginer permet de déposer dans un registre II le code du signe qui est en train de passer devant la pédale. Le contenu de II se renouvelle donc à chaque génératrice qui passe et un cycle, durant lequel défilent les 40 codes, recommence à chaque tour du tambour.

Comment va-t-on se servir de ces éléments pour imprimer le texte de la ligne dont les codes siègent dans le registre I? Suivez bien, c'est vraiment beau. On fait tourner le tambour autour de son axe, de façon que, par exemple, 600 génératrices défilent devant la pédale chaque minute, ce qui exige une rotation à 10 tours par seconde. Ce n'est pas un maximum, on sait aujourd'hui faire tourner des gros tambours à 100 tours/seconde. Le temps τ qui s'écoule entre les passages de *deux* génératrices *successives* devant la pédale vaut donc 1/10 de seconde, temps d'un tour complet, divisé par 40, soit 1/400 de seconde ou 2 500 microsecondes. Rappelez-vous cette valeur de τ.

Voyons comment va s'imprimer sur le papier une certaine ligne dont les 80 caractères ont été déposés par le programme d'édition dans les 80 cases du registre I, à raison d'un code de caractère par case. L'idée simple et lumineuse est de consacrer un tour complet du tambour, c'est-à-dire dans notre exemple 1/10 de seconde, à cette écriture, et de faire demander par chacune des 40 génératrices qui passent à tour de rôle devant la pédale pendant ce tour, et qui porte l'un des 40 caractères possibles, *si le registre I veut de ce caractère et en quelles places.* Si oui, on s'arrange pour que les marteaux qui desservent les dites places soient excités au vol et provoquent donc l'inscription dudit caractère aux dites places. Puis on passe à la génératrice suivante, c'est-à-dire au caractère suivant, on recommence l'appel du registre I pour ce caractère, et celui-ci s'inscrit au vol à toutes les places où la ligne à écrire le demande. Et ainsi de suite avec les 40 génératrices, donc 40 caractères, pendant le tour complet consacré à la ligne. Nous aurons ainsi composé notre ligne, d'une façon amusante, en imprimant d'abord tous les « *a* » dans toutes les places à la fois de cette ligne où doit figurer un *a*, puis tous les « *b* » à la fois, ..., puis tous les « 1 » à la fois, etc., chaque impression de chaque caractère en toutes les places de la ligne où le texte le demande s'effectuant au vol pendant le temps de 1/400 de seconde où la génératrice portant ce caractère est disponible devant la pédale. Le secret était d'abandonner la coutume ancestrale d'écrire une ligne en suivant dans l'ordre les places successives de gauche à droite pour déposer chaque fois le caractère que le texte demande à cette


registre II de la lettre qui passe
tambour
papier
pédale
N°k
N°p
amplificateurs
F_k
F_p
H
amplificateur
comparateur
mélangeur général
E_k
E_p
registre I de la ligne à imprimer
e
N°k
N°p
l'axe
un parallèle
une génératrice
tops ultrarapides
(autant pendant le séjour d'une génératrice qu'il y a de case à visiter dans II)
chaîne à décalage
venant de l'ordinateur

place. Au lieu de cela on écrit simultanément un même caractère dans toutes les places de la ligne où il doit figurer.

Le logicien profond que vous êtes devenu comprendra facilement le circuit logique représenté sur la figure et qui permet d'effectuer cette opération. Plaçons-nous à l'instant (fugitif) où la génératrice qui passe devant la pédale porte, mettons, la lettre $\boxed{e}$, dont le code siège donc au même instant dans le registre II et de ce fait sur le côté droit du comparateur. Supposons par exemple que cette lettre *e* figure aux places N° *p* et *k* de la ligne à écrire. Dès que la génératrice « *e* » est descendue devant la pédale, on démarre la chaîne à décalage douée de 80 maillons. C'est-à-dire qu'on va envoyer sur son entrée une série de 80 tops rapides qui apparaîtront successivement aux 80 sorties *pendant* le séjour de la génératrice « *e* » devant la pédale. Ces tops ultrarapides ouvrent successivement : d'une part les portes E qui reçoivent les codes des lettres siégeant dans les 80 cases du registre I et conduisent leurs contenus vers le côté gauche du comparateur; et d'autre part les portes F autorisant un éventuel signal régnant sur la sortie H du comparateur à exciter un marteau. Le fonctionnement est évident : au *p*-ième top, pendant le séjour de la génératrice « *e* » devant la pédale, la sortie N° *p* de la chaîne à décalage génère du signal, la case N° *p* de I dévide son contenu vers le comparateur qui, constatant que ce contenu est la même lettre « *e* » que celle siégeant à cette époque en II, envoie du signal sur H. Comme au même instant la porte d'accès du marteau N° *p* est ouverte, celui-ci frappe son coup et la lettre « *e* » s'imprime à la place N° *p* de la ligne.

Il en ira de même au cours du même temps τ de séjour de la génératrice « *e* » devant la pédale, lorsque sera récité le *k*-ième top, et on aura encore une impression de la lettre « *e* » à la place N° *k*. A la génératrice suivante, quand ce sera la lettre « *f* » qui sera proposée par le registre II, le même processus fera automatiquement imprimer cette lettre à toutes les places où il doit y avoir un *f* dans la ligne à imprimer.

▶▶ — Mais alors, m'interrompez-vous, les impressions d'une même lettre le long de la ligne ne sont pas *vraiment* simultanées! Elles se suivent à mesure que les tops qui émergent sur les 80 fils de sortie de la chaîne de décalage vont visiter les 80 cases de I?

▶ — C'est vrai. Seulement songez que cette visite nous fait passer sur chacune des 80 cases pendant 1/80 du temps de séjour τ d'une génératrice devant la pédale, lequel temps τ vaut 1/400 de seconde. Le sondage d'une case s'effectue donc en

$$\frac{1}{80} \times \frac{1}{400}, \quad \text{soit} \quad \frac{1}{32\,000}$$

de seconde, c'est-à-dire 30 microsecondes. Cela suffit largement à l'opération *électronique*, mais ne crée aucun retard *perceptible à nos sens* entre les divers instants où la même lettre est imprimée aux divers endroits de la ligne. C'est pourquoi je m'étais permis de dire que ces impressions sont « simultanées ».

▶▶ — Mais les marteaux, eux, ne sont pas électroniques! Ils ne peuvent tout de même pas frapper des coups qui ne durent que 30 microsecondes!

▶ — Certes. Aussi ne le leur demande-t-on pas. Le marteau de la place N° *p* est *invité à frapper* lorsque défile le top N° *p*, mais pour le faire il dispose de *tout le temps* τ, jusqu'au passage de la génératrice suivante, soit 1/400 de seconde. C'est

encore très très peu pour la percussion d'un organe mécanique, mais c'est faisable lorsqu'on rend les marteaux aussi légers et aussi bien équilibrés que possible et qu'on amplifie très fort les signaux qui excitent les solénoïdes à la sortie des portes F. Vous voyez quelle subtile combinaison on fait des deux règnes électronique et mécanique, bouchant les défauts de l'un par les vertus de l'autre.

▶▶ — Si je comprends bien, il faut attendre, pour qu'une ligne soit imprimée, un tour complet du tambour, afin que les 40 génératrices aient toutes eu l'occasion de proposer à l'impression de la ligne les 40 caractères possibles.

▶ — C'est bien ce que je vous ai dit au début : *un* tour complet du tambour, soit 1/10 de seconde, pour *une* ligne. Le mouvement du papier est synchronisé de façon qu'il avance d'un interligne à chaque tour de tambour. C'est bien ce qui nous procure 10 lignes imprimées par seconde, soit 600 à la minute. Ce rythme, il faut le voir réalisé par une imprimante en action pour sentir ce qu'il signifie. Les organes servant à exciter les marteaux sont assemblés en général dans un bloc noir qui couvre le papier d'un bout à l'autre, et ce bloc noir, dans un bavement monstrueux, dépose *chaque seconde* sous lui ***dix*** lignes de 80 caractères complètement imprimés. On a à peine le temps de regarder une avalanche de signes qu'une autre s'installe. En une minute a défilé un demi-mètre de papier tout couvert de texte, dans un bruit ressemblant vaguement à celui que ferait un crocodile à 80 dents mastiquant sa proie, bruit qui émane de 80 marteaux pleuvant en grêle sur le papier. Il vous semble que l'ordinateur n'est en fin de compte qu'une machine à écrire géante qui a mangé ses dactylos, puisque l'impression folle qui en sort est programmée dans son ventre. Remarquez que la typographie ainsi produite est pauvre, très pauvre, avec seulement ses 40 caractères, sans minuscules, majuscules, gras, italiques, variations de type, etc. Cette allure minable de la typographie produite reste l'un des problèmes non encore résolus de l'univers *ordino*. L'autre problème est lié à la fantastique puissance des imprimantes. Elle conduit les programmeurs à commander des impressions très complètes d'une foule de résultats, de sorte que la « réponse » à un problème complexe résolu par l'ordinateur est souvent constituée par quelques mètres de papier replié en accordéon, avec toutes ses lignes remplies d'écritures... Et nous voilà obligés de développer une profession nouvelle, celle de « lecteur des piles de papier imprimées »; car il n'est pas à la portée de tout le monde d'interpréter clairement les torrents de signes que déversent les imprimantes des ordinateurs...

Aussi a-t-on songé à munir l'ordinateur d'autres interfaces plus faciles à maîtriser et dont les présentations soient plus suggestives. Nous allons les examiner.

Les tables traçantes

Dans un certain nombre de cas, on peut représenter l'essentiel des résultats trouvés sous la forme de courbes, seules ou formant des familles, voire de véritables dessins. Dans ces cas il est assez facile de mécaniser la présentation. On dresse une belle vitre plane rectangulaire dont les côtés forment deux axes de coordonnées x et y, et une pointe traçante sur un équipage mobile fait de deux bras dont l'un peut se déplacer parallèlement aux x et l'autre parallèlement aux y. Quand l'ordinateur a trouvé qu'il faut un point à une certaine abscisse x et une certaine ordonnée y,

un programme d'édition sort des trains d'impulsions adéquats vers les deux bras qui sont entraînés vers cet x et cet y, après quoi un top spécial fait descendre la pointe qui marque le point sur la vitre. On arrive ainsi à faire tracer à l'ordinateur sur la table les courbes qui incarnent la solution qu'il a trouvée au problème posé.

C'est très joli à voir fonctionner. Mais c'est lent et limité à une classe particulière de problèmes. En outre, ce n'est qu'un moyen de *sortie* de l'ordinateur, pas un moyen d'*entrée*. L'interface qui suit assume vraiment ces deux rôles, et avec des performances prodigieuses.

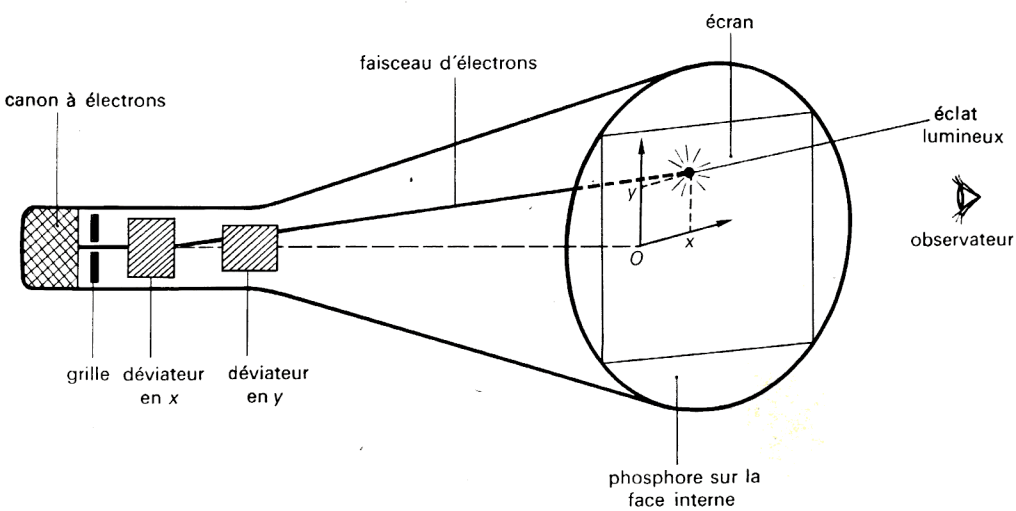
Le tube cathodique

▶ — Savez-vous ce qu'est un « tube cathodique »?

▶▶ — Heu...

▶ — Vous devriez le savoir. Car c'est ce qui remplit les 4/5 (en volume) de votre téléviseur. Et c'est l'une des plus notables conquêtes de l'homme.

Un tube cathodique est une grande cloche de verre ayant, à l'avant, une grosse bombure terminée par un écran, et en arrière un col mince. La cloche est pleine du meilleur vide qu'on puisse faire, et on y exploite deux phénomènes-clés. D'une part on connaît des substances appelées « phosphores », qui se laissent déposer en poudres impalpables et translucides sur du verre, et qui ont la propriété que tout point de la substance qui est frappé par un flux d'électrons émet de la lumière avec une intensité *grosso modo* proportionnelle au flux incident. Cette lumière peut, selon la nature du phosphore, soit *disparaître* aussitôt que le flux d'électrons cesse de tomber, soit *persister* plus ou moins longtemps après. Dans la suite nous n'aurons besoin que du premier type de phosphore, que nous appellerons (s'il faut le spécifier) « instantané ». L'autre phénomène est qu'on arrive à préparer des « faisceaux » d'électrons qui, dans le vide, peuvent se propager un bon bout de chemin, par exemple 50 cm, voire 1 m,



sans se disperser appréciablement en dépit des forces de répulsion que les électrons exercent les uns sur les autres (comme toute paire de charges électriques de même signe). Les faisceaux d'électrons ainsi préparés ressemblent à de minces filaments d'épaisseur constante. Les sources d'où peuvent jaillir de tels faisceaux s'appellent, d'un nom qui leur convient bien, des « canons » à électrons, et je ne vous dirai pas quelles astuces on déploie pour réussir cette artillerie. C'est de la très haute électronique. On réalise des faisceaux qui ont à peine un cinquième de millimètre de diamètre et engendrent donc, quand ils tombent sur un phosphore, de véritables « points » lumineux qu'on appelle des « spots ».

Ce n'est pas tout. Le faisceau d'électrons peut encore être *contrôlé* de **trois** manières. D'abord, si on place à la sortie du canon une « grille » analogue à celle dont je vous ai parlé à propos des « tubes à vide » inventés par Lee de Forest, on peut varier l'*intensité* du faisceau (c'est-à-dire le nombre d'électrons qu'il transporte) en variant la tension appliquée à cette grille, et cela depuis 0 (faisceau éteint) jusqu'à une certaine valeur maximum de saturation. C'est ce qu'on appelle le contrôle d'*amplitude*. Ensuite, on peut disposer au-dessus du col du tube deux bobines à angle droit telles que si on fait passer du courant dans l'une, le faisceau se dévie dans le sens, mettons, des x, et si on fait passer un courant dans l'autre, le faisceau se dévie dans le sens des y. Ces deux bobines s'appellent les « *déviateurs* » et permettent, en dosant convenablement les courants qu'on leur applique, d'envoyer le faisceau en n'importe quel point de la face du tube couverte par le phosphore. Comme les déviations du faisceau sont proportionnelles aux courants qui les provoquent, on peut dire en simplifiant qu'on positionne l'éclat lumineux au point (x, y) de l'écran en envoyant dans les deux déviateurs un signal « égal » à x et un signal « égal » à y.

En télévision les deux déviateurs reçoivent des signaux ajustés de telle façon que le spot décrit des lignes successives de l'écran, de gauche à droite et de bas en haut, comme votre œil quand il lit une page : c'est ce qu'on appelle une « *trame* ». Quand le spot passe par un certain point d'une certaine ligne, le signal appliqué à la grille prend la valeur nécessaire pour que la brillance du spot ait le niveau requis en ce point.

Tout cela se fait à une vitesse fantastique. Comme le phosphore employé est du type instantané, l'impact du faisceau électronique ne produit qu'un flash fugitif, que votre œil serait bien incapable d'intégrer. Il ne conserve le souvenir du flash que pendant un temps très court de l'ordre de 1/20 de seconde. Pour que vous ayez la sensation de voir une scène, il faut donc que le flash se *reproduise* mettons 25 fois par seconde, c'est-à-dire qu'une trame complète doit être parcourue en 1/25 de seconde. Or une trame complète de télévision contient 800 lignes, sur chacune desquelles on veut pouvoir discerner 600 points. Un point est donc visité pendant un temps égal à

$$\frac{1}{25} \times \frac{1}{800} \times \frac{1}{600} \quad \text{soit} \quad \frac{1}{1\,200\,000}$$

seconde ou 0,8 microseconde. Eh bien, le faisceau cathodique dans le vide est en effet capable d'effectuer des mouvements et de recevoir des modulations aussi inconcevablement rapides, et même 10 fois plus! On pourrait, si on voulait, balayer une trame de 1 000 lignes avec 1 000 points discernables sur chaque ligne en un centième de seconde. Le faisceau cathodique tombant sur un phosphore est donc le crayon de très très loin le plus rapide et le plus souple du monde.

Dans l'univers *ordino* on ne reproduit pas, comme en TV, des photos avec des demi-teintes. On *dessine* sur l'écran avec un trait de luminosité toujours la même. Il en résulte qu'on n'utilise en général pas la modulation du faisceau en amplitude, sauf pour l'éteindre ou l'établir ; et, dans ce dernier cas, à niveau constant. On utilise pour les déplacements du spot deux modes différents. Dans le premier mode le faisceau décrit régulièrement une trame TV, par exemple avec une période de 1/25 de seconde, auquel cas on se borne à allumer le spot quand il passe par le point où on veut qu'il y ait de la lumière, et le signal de contrôle est alors appliqué sur la grille. Dans le second mode on laisse le faisceau tout le temps sous sa pleine intensité, et on l'envoie d'un point à l'autre de l'écran par des sauts de puce, le signal de contrôle étant alors la paire de signaux x, y appliquée aux déviateurs. Dans les deux cas, si on veut qu'un dessin tracé par le spot subsiste au regard de l'homme, il faut *l'entretenir, en le renouvelant au rythme de 25 fois par seconde*. Cela entraîne la nécessité de déposer les signaux qui contrôlent le spot dans une mémoire spéciale appelée « *mémoire d'entretien* » et qui débite la même suite de valeurs 25 fois par seconde, jusqu'à ce qu'elle reçoive l'ordre de passer à une autre suite de valeurs, ce qui change le dessin.

Les « consoles » de visualisation

Que va-t-on faire avec ce merveilleux outil ?

Pratiquement tout ce qu'on veut. Sa ductilité n'a pas de limite. D'abord on peut *écrire* avec le faisceau sur l'écran, eh oui, écrire des lettres formant des phrases, tout comme avec une imprimante. Il suffit que le programme d'édition ait déposé en mémoire les coordonnées successives de tous les points qui forment les contours de toutes les lettres à écrire et d'envoyer ces coordonnées sur les déviateurs, pour que le spot suive fidèlement les graphismes formant ces lettres, et cela se passe avec une vitesse telle que vous n'avez pas le temps de voir les lettres se former, les phrases ont l'air d'émerger toutes faites. Il existe même des tubes cathodiques spéciaux, appelés « *caractrons* », qui permettent à chaque lettre d'être effectivement projetée d'un seul coup sur l'écran. Pour cela on complique les éléments de contrôle contenus dans le col du tube. Le faisceau est rendu volontairement assez large pour embrasser la taille maximum des lettres. Dans une première section du col, deux déviateurs envoient le faisceau vers l'une des 64 places d'une *plaque* où les lettres, chiffres et autres signes sont creusés sous forme de fentes profilées. Lorsque le faisceau tombe en une place de la plaque où est entaillée la lettre $\boxed{m}$, par exemple, seuls les électrons qui tombent *sur* la fente dessinant cette lettre peuvent passer, de sorte que la lettre a été en quelque sorte moulée dans le faisceau. Celui-ci est maintenant renvoyé sur l'axe du tube dans une seconde section du col dont les déviateurs reçoivent des signaux complémentaires de ceux qui ont sélectionné la lettre $\boxed{m}$. Dans une troisième section du col, deux nouveaux déviateurs envoient le faisceau, désormais « imprégné » du $\boxed{m}$ en tel point voulu de l'écran. Et quand il y tombe, c'est la lettre $\boxed{m}$ qui s'illumine d'emblée. Un trou central dans la plaque permet au faisceau si on le souhaite, de passer son chemin le long de l'axe *sans* être dévié sur la plaque à caractères, c'est-à-dire jusqu'à la troisième section du col. Dans ce cas un signal spécial permet aussi de rétrécir le faisceau jusqu'à une grande finesse, et le tube est prêt alors pour fonc-

tionner à la manière habituelle, *sans* présélection de lettres, à partir de la 3e section du col.

Bien entendu, le faisceau peut dessiner n'importe quoi d'autre que des lettres, selon ce que lui ordonne le programme d'édition : des courbes, des droites, des flèches, des pointillés. Et bien entendu les informaticiens, gent habile, n'instruisent pas le faisceau lourdement en lui fournissant la suite des coordonnées de tous les points qu'il a à suivre. Doit-il tracer un arc de cercle, ils font mettre dans le programme d'édition seulement les coordonnées du centre et des deux extrémités avec l'ordre « cercle », et ce programme déclenche alors automatiquement un « *sous-programme de dessin* » qui engendre l'équation mathématique du morceau de ligne à tracer, et asservit le spot à rester sur des points de l'écran qui satisfont à cette équation. Avec une vitalité prodigieuse, l'industrie de l'informatique, en quelques années après la première apparition du tube cathodique comme organe de sortie, a produit de véritables « *langages de dessin* » qu'on vend tout faits et dans lesquels des dizaines d'éléments de graphisme sont préparés, « tournables », « translatables », gonflables ou rétrécissables à volonté, et qu'on peut appeler par des ordres spéciaux prévus à cet effet. Ces « dessins patrons » sont des sortes de « super-lettres » dans un vaste « alphabet pictural », avec lesquelles on peut composer toutes les figures qu'on veut. On a en outre élaboré des « *programmes de composition* », analogues aux « programmes d'assemblage » dont je vous ai parlé au Chapitre 2 qui examinent le programme d'édition avant qu'il éclate en une série de sous-programmes de dessin et regroupe ceux-ci pour les faire intervenir aux moments où le dessin correspondant peut se déployer avec un maximum d'économie, c'est-à-dire par exemple qu'on trace ses lignes en remplissant l'écran de gauche à droite ou du bout au centre et non en sautant sans ordre d'une zone à l'autre. C'est en somme un « *programme de gestion* » des graphismes, ou si vous voulez du faisceau. Nous en sommes donc à quatre : *le programme d'édition*, *le programme d'entretien* (vous l'avez peut-être déjà oublié, mais il faut toujours renouveler les dessins 25 fois par seconde), *les sous-programmes de dessin* et *le programme de gestion*. Tout ce software, plus ou moins préfabriqué, s'incarne dans de la circuiterie classique d'ordinateur (commutation et mémorisation) qui est en général logée dans le même meuble que le tube cathodique. Ce meuble devient de ce fait une véritable « console » qu'on appelle « de visualisation », encore que cette épithète tende à devenir inutile. La console est montée de façon qu'un usager puisse s'asseoir devant l'écran et bien jouir du spectacle qui s'y forme.

Songez que les dessins, obéissant à des équations ou à des ordres *calculés* dans l'ordinateur, peuvent avoir une exactitude, une précision, une richesse qu'aucun artiste graphique ne saurait atteindre. Par exemple, les traits découverts et cachés dans une perspective peuvent (si on a prévu les calculs correspondants dans le programme) s'arrêter, reprendre, se chevaucher avec une fidélité telle qu'on croit vraiment voir la 3e dimension, le relief et la perspective. Et des textes, dessins, cartes, schémas complets paraissent et disparaissent devant vos yeux avec une telle promptitude qu'on les dirait évoqués ou chassés par quelque fée. On peut aussi instruire le programme de gestion de façon qu'il « meuble » la figure à un certain rythme, faisant d'abord apparaître quelques grandes lignes, puis des détails, puis des ombres, puis des cotes ou légendes, etc. Plusieurs sociétés cinématographiques se sont constituées pour créer des bandes de dessins animés sur ces consoles, à partir d'ordinateurs. Certaines de leurs réalisations vous coupent le souffle.

Le mode conversationnel

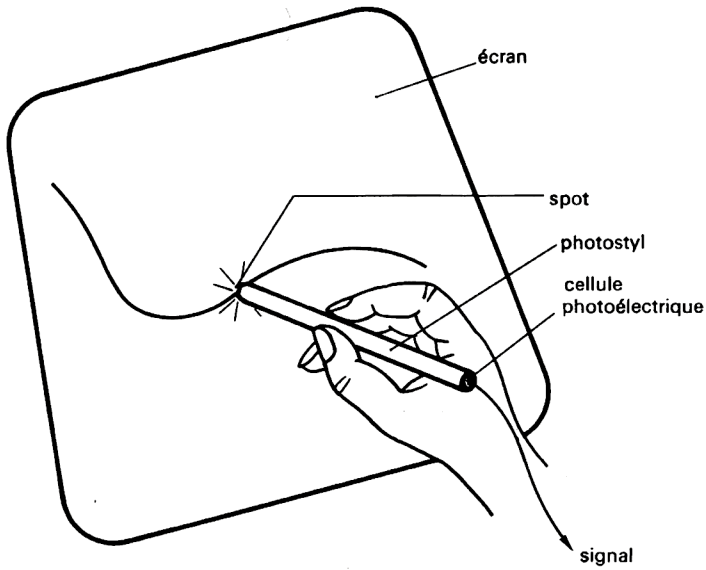
Mais ce n'est pas tout. Jusqu'ici la console n'a servi qu'à afficher des textes ou figures incarnant les solutions trouvées par l'ordinateur aux problèmes qui lui avaient été posés. Ce n'était donc, comme la table traçante, qu'un *organe de sortie*. Or on peut aussi en faire un *organe d'entrée*, et cela d'une manière telle que l'usager peut véritablement « converser » avec l'ordinateur. Vous allez tout de suite comprendre ce que cela veut dire.

D'abord un brin de technique. Je ne vous ai pas parlé jusqu'ici d'un interface qui est pourtant le plus nécessaire et le plus simple de tous, le **clavier** constitué par ces touches, boutons ou manettes qu'on trouve comme organe de commande sur le tableau de bord de toutes les machines électromécaniques ou électroniques. Chaque touche qu'on enfonce ouvre ou ferme certains circuits et provoque des effets déterminés. Tout ordinateur dispose de vastes claviers qui permettent de susciter des événements variés, comme copier tel segment d'une bande, ou introduire telle donnée, ou faire consulter tel sous-programme, etc. On a en particulier besoin d'un tel clavier pour « initier » un ordinateur quand il est entièrement vide et mort, car de ce fait même il ne contient aucun programme enregistré capable de le faire marcher, et il faut lui enseigner les premiers pas comme à un enfant, au clavier manuel, meublant quelques cases de mémoire et démarrant un vade-mecum de transition pour que la machine sache au moins absorber un premier bout de pré-programme, lequel une fois parcouru pourra provoquer l'appel des programmes définitifs plus complets. Donc, une console de visualisation contient d'abord un clavier de ce genre qui permet de rentrer dans l'ordinateur, par enfoncement manuel des touches, des données, des questions ou des ordres qui sont inscrits en clair sur des voyants. C'est dans la console même que l'effet qu'on veut susciter par une touche est codé en langage machine.

Un second canal par lequel peut s'effectuer aussi une certaine forme de *retour* de la console vers l'ordinateur est constitué par un ravissant petit crayon terminé par un fil électrique souple et qu'on appelle vulgairement « crayon électronique » et académiquement, « **photostyl** ». Il a en effet l'allure d'un crayon, mais il est creux. Sa pointe, qui est placée sur l'écran cathodique, est percée d'un trou et au fond du crayon creux se trouve une cellule photoélectrique.

Si je pose le crayon sur l'écran en un point que l'impact du faisceau fait reluire, la cellule du fond émet un signal, que je veux utiliser comme marque du fait que le spot vient de passer en ce point-là.

Et quel est ce point ? Ma main, mon œil le savent, mais pas l'ordinateur. Comment sera-t-il informé ? Il y a pour cela plusieurs procédés. Si le spot balaye l'écran selon une trame sous-jacente, la solution est immédiate car dans ces balayages les points de l'écran sont visités à tour de rôle dans le temps, et l'*instant* où le signal sort du photostyl suffit à identifier le point où le crayon fut posé : il suffit pour envoyer ses coordonnées (x, y) à l'ordinateur, que deux compteurs débitent perpétuellement les valeurs de x et de y à mesure que le faisceau parcourt la trame et qu'on admette leurs sorties vers l'ordinateur lorsque le photostyl débite son flash : ce sont automatiquement les coordonnées du point où il a été posé et qu'un spot a visité. Je peux ainsi *indiquer un point de dessin* à l'ordinateur simplement en le touchant avec le photostyl.



Si le faisceau balaye l'écran dans le mode « saut de puce », l'identification d'un point où il a fait impact et où j'ai posé le photostyl est plus difficile, mais on peut encore y parvenir de diverses manières. Par exemple, on peut tendre en travers de la face avant deux réseaux de fils métalliques très fins, à peine visibles, les uns parallèles à l'axe des x et les autres parallèles à l'axe des y, réseaux normalement isolés entre eux par une très mince couche d'isolant transparent. Cela n'empêche pas de voir la figure affichée, mais quand on pose le photostyl sur un croisement d'un fil x et d'un fil y, un effet dit « capacitif » (analogue à celui qu'on exploite pour vous épater dans les ascenseurs dont vous pouvez régler la course rien qu'en effleurant une touche de votre doigt) fait que du signal passe sur ces fils et communique ainsi les coordonnées voulues. On peut aussi employer une astuce assez vicieuse qui évite de surcharger l'écran avec les fils. Cela consiste à réserver une petite fraction dite « temps de service » de chaque période de répétition du dessin, période qui, vous vous en souvenez, vaut 1/25 de seconde, pour envoyer le faisceau pendant ce très petit temps dans une position spéciale égalemeent dite « de service » et bien repérée. On commence par poser le photostyl sur ce « spot de service » puis on déplace le photostyl à partir de là vers le point qu'on veut couvrir en *asservissant* le spot de service à le suivre au moyen de signaux déviateurs tirés du mouvement même du photostyl et appliqués en retour aux déviateurs pendant les « temps de service ». Les coordonnées des points parcourus par le spot pendant cette course sont connues, et comme son point de départ est connu, les coordonnées de son point d'arrivée, c'est-à-dire du point qu'on veut viser et désigner à l'ordinateur, sont également connues de l'ordinateur. Dans une variante plus radicale de cette solution on abandonne tout à fait le photostyl et on le remplace par ce qu'on appelle un petit « manche à balai » qu'on déplace à la main et qui agit sur les déviateurs pour déplacer en conséquence le « spot de service », comme on veut, de gauche à droite et de haut en bas. Les coordonnées du « spot de service » sont donc perpétuellement connues, et quand on l'a amené en regard

du point du dessin qu'on veut viser et désigner, on enfonce du pouce un petit bouton qui a pour effet de débiter les coordonnées atteintes à ce moment vers l'ordinateur.

Au total, vous constatez qu'on dispose d'un moyen pour *viser-désigner tout point du dessin.* C'est un « *pointeur géométrique* », alors que le clavier nous donne un « *moniteur logique* ». Et la combinaison des deux présente des possibilités de retour de l'homme vers la machine d'une richesse et d'une commodité inouïes. Supposez, par exemple, que j'aie demandé à l'ordinateur de résoudre les équations qui déterminent la flèche d'un pont à ériger, et que je l'aie programmé pour qu'il me conte sa solution en image en faisant s'afficher la silhouette du pont sur ma console. La voici devant moi. Maintenant je me pose des questions : et si une tension spéciale venait à s'exercer en tel point sur tel arc-boutant, de combien changerait la tension au milieu? Je désigne le point au moyen de mon photostyl, et par le clavier je compose à l'usage de l'ordinateur l'ordre de reprendre les calculs avec les nouvelles hypothèses. Comme il ne s'agit que d'une passe d'opérations déjà faites, l'ordinateur l'aura achevée avant que je puisse dire ouf, et si j'ai préparé en conséquence mon programme d'édition, la valeur de la tension au milieu est tout de suite là, sous mon photostyl, écrite en clair sur le point médian du pont. Je suis vraiment le sorcier qui évoque instantanément sous sa main l'image que je veux. Autre exemple : je suis banquier et je me fais afficher sur la console la suite des opérations de retrait ou dépôt effectuées par mon client Tartempion qui a un compte chez moi. Cette suite a été établie par mon ordinateur. Subitement, je veux en savoir plus sur telle opération : au profit de qui a-t-elle été faite, par quelle autre banque a-t-elle transité, etc. Je *désigne* avec le photostyl la ligne où figure cette opération, j'ordonne via le clavier à l'ordinateur de me chercher les renseignements complémentaires et de me les envoyer, et zoup, les voilà écrits dans un coin de l'écran.

Et ainsi de suite sans fin. Réalisez-vous la nature et l'intérêt de la nouvelle puissance que nous venons d'acquérir? C'est celle de ***converser*** avec l'ordinateur. Jusqu'ici en effet les rapports entre l'homme et la machine conservaient ceci de guindé, de restrictif, que la machine ne fournissait à l'homme qu'une réponse à la fois, puis se taisait. Or l'homme pensant cherche sa voie par des successions de tâtonnements, d'essais, d'approfondissements, de spécialisations et de généralisations, qui reposent sans cesse les problèmes que l'ordinateur a résolus sous de nouveaux angles, ou en y découpant des tranches. Faudra-t-il, pour avoir les réponses à ces coups de sonde lancés dans divers azimuths, refaire à chaque fois un programme complet et demander une sortie imprimée de la réponse? Ce serait très lourd, très cher, très long. Le fil de la pensée se romprait, l'inspiration se perdrait. Le processus idéal est tout autre, et il est devenu possible grâce aux consoles à tubes cathodiques. Ce processus consiste à prévoir d'avance toutes les quêtes partielles qu'on pourrait aimer faire autour du problème sur le vu d'une première réponse, et de préparer la machine de façon qu'elle puisse les parcourir sur demande et répondre à chacune assez vite pour que ça ait l'air « du tac au tac » ... tac au tac à l'échelle humaine s'entend. On exploite ici à fond l'énorme abîme qui existe entre l'échelle *homo* et l'échelle *ordino*. Car pour la première une réponse est du « tac au tac » si elle vient dans quelques secondes, tandis que pour la seconde cet « instant » a suffi pour que s'exécutent un ou deux millions d'opérations...

Les consoles de visualisation ont donc permis d'établir entre l'homme et l'ordinateur ce mode de collaboration plus voisin des démarches normales de l'esprit qu'on

appelle « *conversationnel* ». Et la « conversation » ainsi devenue possible peut être infiniment plus fertile que celle que le chercheur mène avec son propre esprit ou avec des collègues. Il peut en effet organiser infiniment plus d'essais, explorer jusqu'au bout infiniment plus de chemins, conserver en mémoire infiniment plus d'aperçus et en outre, il reçoit, des figures mêmes qui s'affichent sur l'écran, infiniment plus de stimulations mentales. Je considère quant à moi que la console de visualisation est aujourd'hui l'interface suprême, et que sa mise au point a constitué le progrès le plus important accompli en informatique dans les dernières années. Vous allez la rencontrer souvent, et chaque fois avec un plaisir artistique aussi bien que scientifique, dans la troisième partie de cet ouvrage consacrée aux applications des ordinateurs.

On a passé en revue la vaste galerie des accessoires appelés « *périphériques* », qui permettent à l'homme de communiquer avec l'ordinateur et vice-versa, et que pour cette raison on appelle aussi des « *interfaces* ». Vous avez rencontré ainsi successivement les célèbres *cartes perforées,* où les tops et tips qui constituent l'information codée se matérialisent par des trous ou pleins, et qui se prêtent à de multiples jeux de tri, recopie, interclassement; le *ruban perforé* où les trous et pleins sont faits sur des tranches successives d'un ruban de papier qui défile; la *bande magnétique,* où les tops et les tips sont de microscopiques aimantations imprimées sur une poudre magnétique enduisant une bande, et qui, grâce à des astuces mécaniques remarquables, permet d'emmagasiner des millions de digits sans aucune équivoque. Cela en fait une mémoire dite « *séquentielle* » où l'accès à une information exige un mouvement mécanique, par opposition aux mémoires à tores statiques qui permettent d'accéder instantanément et discrétionnairement à n'importe quelle case, et qui, pour cette raison, sont (improprement) appelées à « *accès aléatoire* ». Le temps qu'il faut attendre avec une bande pour voir arriver l'information demandée peut atteindre quelques minutes. Ce temps s'abaisse à une petite fraction de seconde quand on dispose l'information sur un *tambour magnétique* tournant, ou sur une pile de *disques magnétiques* qui, malgré leur plus grande vitesse d'accès, ont une capacité égale ou supérieure à celle des bandes mais sont moins maniables. Là-dessus vous avez demandé aussi comment on peut rendre visibles et lisibles les résultats élaborés par l'ordinateur, et on vous a présenté les merveilleuses *imprimantes,* combinaisons d'ultra-mécanique et d'électronique capables d'imprimer 600 lignes de 80 caractères par minute; puis le plus merveilleux encore *tube cathodique,* où un faisceau d'électrons commandé par l'ordinateur dessine sur la face du tube les réponses aux questions posées, avec une vitesse telle que les graphismes paraissent surgir tout entiers instantanément. Et vous avez vu, pour terminer, qu'on a des moyens pour retourner de ces « *consoles de visualisation* » vers l'ordinateur, ce qui institue entre l'homme et son outil le type d'échanges aller et retour qu'on appelle « *conversationnel* ».

8

Un peu d'histoire et un peu de prospective

Le passé

Beaucoup d'auteurs ont un faible pour le sport consistant à déterrer des antécédents curieux, qui apparaîtront d'autant plus étonnants qu'ils remontent plus loin. J'en connais qui vendraient leur âme s'ils pouvaient découvrir un grimoire oublié prouvant qu'un alchimiste scholaste sorcier du Moyen Age avait déjà songé à la fission de l'atome. Personnellement, je n'ai pas beaucoup de goût pour cette pente d'esprit. Il n'est pas de domaine scientifique dans lequel on ne puisse découvrir des précurseurs si on donne à ce mot une élasticité suffisante. Mais la quête de ces signes prémonitoires lointains est en général stérile, voire trompeuse. On peut dire par exemple que le boulier des Romains est le précurseur de l'ordinateur moderne parce qu'on y fait des calculs avec des objets. Mais il est aussi vrai, et même plus vrai, de le considérer comme un obstacle dressé sur la découverte d'un ordinateur, car il a détourné l'esprit de bien d'autres éléments fondamentaux de l'ordinateur tels que le codage, la mémoire, la programmation.

Je n'en déduis nullement qu'il est inutile de scruter l'histoire des idées et des techniques. Au contraire. C'est, je crois, l'un des domaines d'étude les plus passionnants, les plus riches d'enseignements sur les mécanismes de l'esprit humain, individuel et collectif. Seulement ces enseignements vont souvent *à l'encontre* de l'optique si répandue qui consiste à s'ébaubir devant les intuitions géniales qui auraient animé certains chercheurs incompris des siècles avant que leurs idées fussent reprises et conduites à terme. Les études historiques sereines montrent en général que lorsque ces intuitions n'ont pas abouti, c'est qu'elles étaient en réalité fausses, ou bornées par quelque côté, ou engoncées dans tant de considérations aberrantes empruntées aux vues limitatives ou aux techniques archaïques de l'époque, qu'elles devaient nécessairement conduire à des impasses. Chaque fois que je lis les travaux même des plus grands novateurs, je suis frappé, au moins autant que par leur audace, par ces multiples idiosyncrasies intellectuelles ou technologiques qui les ligotaient ou les alourdissaient, par ces conceptions adventices souvent si alambiquées qu'on se demande comment des éclairs ont quand même pu jaillir à travers leur pâte. Et c'est précisément ce séculaire, ce gigantesque travail tellurique par lequel l'esprit se dégage lentement de ses entraves, qui constitue à mes yeux la leçon prodigieuse de l'histoire... ce qui suppose qu'on reconnaît les entraves. D'en briser une seule requiert déjà assez de génie pour qu'il n'y ait pas besoin d'ajouter que ce génie avait tout vu. Bref, *les créateurs de l'objet moderne sont en général ses créateurs authentiques.*

C'est pourquoi je ne vous ferai pas défiler la galerie coutumière des ancêtres de l'ordinateur, bien que parfois, comme avec la machine à calculer de Pascal, ou les automates de Vaucanson, ou les cartes perforées de Jacquard, ils présentent des traits remarquables et amusants. La première réalisation dont on peut dire qu'elle a quelque ressemblance foncière avec notre ordinateur est la machine à calculer réalisée au milieu du siècle dernier par l'Anglais Babbage. Tout s'y fait par des engrenages mécaniques, c'est-à-dire avec une lenteur qui rendait la performance peu utile, mais on y trouve déjà, sinon le code binaire, du moins l'idée d'un tout petit bout de programme enregistré d'avance sur des cartes perforées à la Jacquard et qui permettait de laisser s'effectuer une suite d'opérations sans que l'homme ait à intervenir à chaque pas. Le rudiment d'une mémoire était là aussi, sous forme de roues dentées qui s'arrêtaient sur la dent correspondant au chiffre qu'il fallait enregistrer. Les informations entraient par des cadrans manipulés à la main, et sortaient sur un papier que marquaient des marteaux porteurs de signes et lâchés en détendant des ressorts. En dépit de la lourdeur de ces mécanismes, Babbage, poussé par une magnifique passion et le soutien de quelques admiratrices, réussit à emmagasiner dans sa mémoire *mille* nombres de cinquante chiffres décimaux!

C'est surtout, dans ce cas, la technologie qui n'était pas prête, il manquait à Babbage l'électronique. Elle manqua aussi au promoteur du second grand pas, l'Américain Hollerith qui, pour aider la statistique démographique américaine, inventa et breveta en 1885 le codage des informations sur les cartes perforées telles que je vous les ai décrites, et les principaux outils pour manipuler ces cartes. On y reconnaissait l'existence ou l'inexistence d'un trou par le fait qu'un balai qui palpait le carton établissait ou n'établissait pas un circuit électrique, et non comme aujourd'hui par le passage ou non d'un rayon lumineux admis sur une cellule photoélectrique, car celle-ci était inconnue à l'époque. Mais, à ceci près, les principaux traitements de cartes imaginés par Hollerith se retrouvent encore dans les tabulatrices et trieuses d'aujourd'hui. Hollerith utilisait l'électricité en plus de la mécanique, mais ignorait toujours l'électronique. Pleinement conscient de l'importance de ses réalisations, il fonde en 1895 pour les exploiter une société qui, en 1911, devait devenir l'illustre « International Business Machine », universellement connue par son sigle IBM.

Grâce à IBM, le traitement dit « électromécanique » de l'information, c'est-à-dire par les machines à cartes perforées, était déjà très répandu lorsque dans les années 1930-1940 s'étendit l'électronique. Ces machines électromécaniques exécutaient surtout des travaux *logiques* sur les cartes, comme classer, dépouiller, trier selon un critère, mais n'étaient pas commodes pour effectuer les opérations *arithmétiques*. Celles-ci ne s'exécutaient que dans des appareils spéciaux, les petites calculatrices de bureau, elles aussi électromécaniques, et qui travaillaient seulement au vol (une opération à la fois posée à la main par enfoncement de touches) parce qu'elles n'avaient pas de mémoire capable d'enregistrer un programme. Or un problème militaire capital requérait qu'on puisse exécuter des séries de calculs programmés dans des temps très courts : c'était la conduite automatique du tir sur cibles mobiles, qui devait permettre de pointer une pièce d'artillerie sur une position *future* de la cible, la position qu'elle aura quand le projectile l'atteindra; et cette position future ne pouvait évidemment être que *calculée* en *extrapolant* le mouvement observé de la cible. D'où la nécessité d'effectuer très vite des séries complexes de calculs.

C'est surtout ce problème qui anima les trois grandes équipes de précurseurs

immédiats de nos ordinateurs et les poussa à se lancer dans des machines électroniques. Ce furent : — en Allemagne, *Friedrich Zuse*, qui réussit son calculateur en 1942, mais se heurta une fois de plus à l'aveuglement des autorités hitlériennes en matière scientifique; — à Harvard, aux USA, *Aiken* qui acheva en 1944 le « Mark I », combinant encore des lampes électroniques avec des relais, qui additionnait en un tiers de seconde et multipliait en 4 secondes; — et *Eckert et Mauchly* qui, en 1946, à l'université de Pennsylvanie, dévoilèrent le premier calculateur tout électronique de l'histoire, appelé ENIAC, ce qui veut dire « Electronic Numerical Integrator and Automatic Calculator ».

Ce monument mérite qu'on s'y arrête quelques instants, car il marque le vrai démarrage de l'informatique moderne. Il reste aussi jusqu'à ce jour la plus gigantesque des machines réalisées. Il pèse trente tonnes et contient 20 000 lampes et 50 000 commutateurs. Il consomme l'énergie électrique d'un gros train, et nécessite donc une centrale spéciale pour l'alimenter et une forêt de ventilateurs pour le refroidir. Cent cinquante hommes doivent le surveiller en permanence pour s'assurer qu'il marche sans hommes... Au départ, le temps de bon fonctionnement entre pannes était de quelques minutes, ce qui n'empêchait pas les foules de s'émerveiller car pendant ce temps le dragon effectuait un million d'additions (cinq mille par seconde). Ce dinosaure fonctionne toujours (!), mais il est désormais relégué dans un musée.

C'est pour l'améliorer que ses créateurs, Eckert et Mauchly, demandèrent à von Neumann l'étude d'où devait sortir le fameux « plan » dont je vous ai longuement parlé, et qui s'incarna, deux ans après l'ENIAC, dans une seconde version moins monstrueuse mais plus astucieuse qui s'appela « EDVAC » (Electronic Discrete Variable Automatic Calculator). Ce n'est qu'à partir de là que le codage binaire de l'information prend un rôle central.

Après la sortie de l'EDVAC, c'est-à-dire dès 1948, les réalisations prennent un rythme tel qu'il devient impossible de les suivre une à une. L'Amérique, avec les grandes compagnies IBM, Remington Rand, RCA, conserve un leadership très prononcé, mais l'Angleterre se met vite dans la course avec la Société Ferranti et la mise au point par Williams d'une mémoire de 4 000 cellules pouvant être visitées discrétionnairement en *un* seul temps machine (les cellules étaient des spots sur la face d'un tube cathodique). La France, avec les sociétés SEA et Bull, n'arrive dans la lice que plus tard et plus modestement, l'Allemagne étant exclue de la course jusqu'en 1958 en raison d'une interdiction des alliés consécutive à la guerre.

On a coutume de parler de « générations » d'ordinateurs. La première serait celle des ordinateurs équipés de *tubes à vide*, la seconde serait marquée en 1956-58 par le remplacement de ces tubes par des *transistors*, la troisième par l'entrée en scène, vers 1965, des circuits dits « *intégrés* » qui assemblent un grand nombre d'étages à transistors directement sculptés au jet de gaz dans un substrat unique formé par un gros morceau de cristal « parfait » de silicium. Personnellement, je ne suis pas très d'accord avec cette classification, parce qu'elle ne fait entrer en ligne de compte que la technologie de la partie « commutation » d'un calculateur, et non la partie « mémorisation », ni la partie « software ». A mon sens il y a en effet un tournant décisif dans les années 1956-58, marqué non seulement par l'avènement du transistor, mais aussi par l'avènement des grosses mémoires à ferrites précontrôlées, et par l'apparition de l'idée des langages de programmation. Ces trois innovations, également importantes, ont vraiment donné à l'ordinateur, à partir de 1958, la fiabilité, la vitesse, la souplesse,

la commodité d'emploi qui en ont fait un objet industriellement utilisable et vendable. A partir de là, le pari était gagné. Les progrès ont continué bien sûr, dans tous les sens et très rapides, mais n'apportant que des améliorations graduelles, sans bouleversement profond. Il n'y a pas de révolution, quant aux fonctions qu'on peut assurer, lorsqu'au lieu d'assembler des transistors séparés sur des plaquettes de circuits imprimées, on les taille dans le sein même d'un bloc cristallin. Le volume en est certes divisé par cinq ou dix, mais seulement pour la partie de la machine qui contient les circuits logiques et de commutation, laquelle partie n'occupe plus de toutes façons que le dixième du volume total. D'autres caractéristiques progressent grâce aux circuits intégrés : la vitesse passe de, mettons, 1 à 0,5 microseconde pour le temps machine, la durée de bon fonctionnement entre pannes passe de, mettons, 1 500 heures à 2 500 heures, le prix baisse de 50 %, on commence à entrevoir la possibilité de fabrications *modulaires*, etc. Simultanément les mémoires se gonflent, on introduit les piles de disques, les tambours géants et les bandes torrentielles. Tout cela est très bon à prendre, cela étend les applications et renforce les assises de l'ordinateur, mais cela ne fait pas une vraie révolution. Les progrès constants du software après 1958 sont d'ailleurs plus importants encore que ceux de la technologie. L'élévation de la programmation au rang d'une discipline scientifique, les langages universels ou spécialisés avec leurs compilateurs, le software de service qu'on vend désormais en même temps que la machine et grâce auquel chaque mise en train d'une installation n'est plus une aventure épuisante, la multiplication des bibliothèques avec sous-programmes préfabriqués et banques de données, tout cela s'inscrit dans une marche en avant continue, puissante, rapide, sur tous les fronts à la fois, mais où on serait bien en peine de marquer une date ou un événement uniques qui auraient changé la face des choses au point qu'un bon spécialiste de 1960 ne reconnaîtrait plus son univers. Même l'introduction accélérée, à partir de 1968, des consoles à tubes cathodiques permettant le mode conversationnel, où je vous ai dit que je vois pour ma part un progrès majeur, n'a pas aussi profondément bouleversé le panorama que le firent le « plan von Neumann » en 1946 et le couple transistor-ferrite en 1956-58.

L'avenir

Y a-t-il en gestation des nouveautés technologiques qui pourraient vraiment apporter une révolution ?

Du côté des circuits logiques ou de commutation, on ne voit pour l'instant se développer rien d'autre d'important que l'intégration toujours plus vaste de circuits toujours plus rapides, plus fiables et moins onéreux dans des blocs monolithiques, ou plutôt sur des plaquettes de « cristaux parfaits » de silicium. On a appris à y inscrire non seulement des transistors ordinaires du genre dont je vous ai parlé, mais encore des assemblages plus savants dits « à effet de champ » et qui reproduisent tout à fait les performances des tubes à vide. Dans cette dernière technologie, appelée « MOS », les différents compartiments qui créent entre eux les jonctions sont formés non plus en dopant le cristal support, mais en y fondant des oxydes métalliques — d'où le signe « MOS » qui veut dire « Metal Oxyde Semiconductors ». Cette broderie sur cristal exige des jets de gaz, des mouvements de micropistolets, des évaporations sous

masques, si fins et si complexes qu'il faut faire diriger le processus de fabrication... par un autre calculateur! Ces circuits sont si compacts qu'une bascule complète tient nettement moins de place que le plus petit des tores de ferrite. Comme la bascule est un élément de mémoire — et non destructive! —, il devient possible d'assembler non plus seulement des registres, mais des mémoires complètes, avec de tels circuits comme cellules au lieu de tores de ferrite. On y gagnera surtout une nouvelle augmentation de la vitesse, car ces circuits peuvent commuter et basculer en 0,1 microseconde, alors que les tores exigent 1 microseconde.

Il n'y a d'ailleurs pas beaucoup de sens à pousser la vitesse de la logique si on ne peut y adapter la vitesse de la mémoire. On produirait l'équivalent d'un orateur qui saurait articuler un mot en deux secondes, mais mettrait une minute à se rappeler quel mot. C'est pourquoi la recherche technologique est plus active dans le domaine des mémoires que dans celui de la logique. En dehors de l'usage des bascules électroniques MOS pour former la mémoire, on pousse très activement ce qu'on appelle la technique des « *films magnétiques minces* » qui permettrait de remplacer le tore, si difficile à tisser, par des petits confetti magnétiques pleins, déposés au jet sur un plan et susceptibles eux aussi de présenter deux états magnétiques très contrastés entre lesquels les passages suivent les mêmes lois que chez les ferrites. On peut s'arranger de manière à remplacer les quatre fils qui doivent *traverser* le tore par des fils déposés *par-dessus* ces confetti selon des axes adéquats, fils séparés les uns des autres par d'impalpables couches isolantes.

Cette technique ne fait gagner ni en vitesse, ni en compacité, mais quand elle sera au point, elle permettra de remplacer le tissage manuel par une fabrication industrielle des plans de mémoire, plus rapide, plus sûre et moins coûteuse. On s'est longtemps demandé si elle serait jamais au point, jusqu'à ce qu'en 1968 on se soit aperçu que beaucoup des problèmes et difficultés qui affligeaient les films minces déposés *à plat* disparaissent lorsqu'on les dépose sur des *cylindres*. Cela permet en particulier de diviser l'épaisseur de la couche par cinq, descendant jusqu'à 2 microns. Du même coup, le basculement de l'état magnétique dans chaque confetti devient plus rapide, et on arrive à des durées de l'ordre de 0,1 microseconde compatibles avec les durées des organes logiques réalisés selon les derniers types des circuits intégrés. On verra ainsi apparaître une génération MOS — films minces, ou « tout MOS », qui multipliera les performances actuelles par un facteur 10 en vitesse et un facteur 5 en prix, encombrement et fiabilité. Mais cette génération n'aura pas encore franchi la barrière de la *capacité* dans la partie statique de la mémoire, de sorte qu'on continuera à avoir besoin des périphériques actuels, c'est-à-dire de systèmes séquentiels, pour la fonction qu'on appelle « mémoire de masse ».

On sait bien, en principe, comment on pourrait loger des milliards de digits dans des espaces réduits. C'est par la photo, car le film photographique moderne de haute résolution n'a plus qu'un « grain » de l'ordre de *quelques microns*, et permet donc d'inscrire sous forme de spots impressionnés ou pas impressionnés plusieurs centaines de digits par millimètre de pellicule, soit un milliard sur une plaque de 10 cm × 10 cm. C'est donc la lumière qui permettrait d'atteindre, et de très loin, la plus grande densité d'écriture. Malheureusement, jusqu'à ces derniers temps, cette lumière était très revêche pour les *déplacements* : on savait produire des rayons ultrafins dont la section n'a que quelques microns, mais on ne savait pas les envoyer ici ou là, donc pas *explorer* la pellicule photo, et, par suite, pas aller lire le contenu d'une adresse désignée,

autrement que par des jeux de miroirs, c'est-à-dire des mouvements mécaniques incapables de tenir la course *ordino*. Depuis qu'on sait produire ces mêmes rayons ultrafins avec une lumière non plus cafouilleuse — je parle de sa structure spectrale — mais cohérente — j'ai cité les rayons laser — on a l'espoir de pouvoir dévier ces rayons, pour visiter les micro-taches faites sur la photo, au moyen de signaux électriques, c'est-à-dire avec l'ultra-vitesse et la précision qu'exige l'exploitation d'une mémoire *ordino*. C'est l'une des pistes notables vers une mémoire de masse statique, non séquentielle, c'est-à-dire une mémoire dont toute adresse est accessible discrétionnairement en un seul temps machine. Ça donnerait lieu à un ordinateur sensationnel où l'électronique fait marcher de l'optique.

Si cette piste aboutit, on aura vraiment franchi un grand tournant. On en aura franchi un encore plus significatif si on parvient à changer le *mode* de consultation actuel des mémoires. Réfléchissez que ce mode n'est pas si satisfaisant que ça. Il oblige en effet à *adresser d'abord et lire (ou écrire) ensuite*, sous le couvert de l'adresse, dans le seul casier que celle-ci désigne. Comme ce serait formidable si on pouvait *inverser les termes* et, pour la lecture par exemple, annoncer d'abord une information, puis voir s'ouvrir tous les casiers qui la contiennent. Pour l'écriture, ce mode « inversé » consisterait à proférer d'abord l'information à écrire, et à voir surgir ensuite les casiers vides capables de l'enregistrer, et qui diraient leurs adresses. Lorsque, par exemple, on doit trier un fichier selon un critère, c'est-à-dire détecter et sortir toutes les fiches qui portent un certain mot, il faut aujourd'hui faire défiler tout le paquet, examiner pour chaque fiche si elle contient ou non ce mot et, si oui, ouvrir une trappe pour grouper ensemble toutes celles qui le contiennent. Il s'agit d'un vrai défilement mécanique lorsque les fiches sont inscrites sur des supports séquentiels, ce qui est le cas général quand il y en a des masses. Mais même si leurs informations avaient pu être toutes inscrites dans la mémoire statique, il faudrait encore faire défiler une à une les *adresses* de celle-ci pour lire coup par coup ce que chacune tient emmagasinée. Au contraire, si on avait le pouvoir d'un sergent qui, criant « Visite! », voit sortir aussitôt des rangs tous les soldats désireux de se faire examiner à l'infirmerie, il suffirait d'afficher le mot servant de critère, et tous les casiers contenant ce mot viendraient déverser leurs contenus simultanément, en un seul temps machine. Une mémoire interrogeable de cette façon, par *contenu* et non plus par *adresse*, s'appelle « *associative* » et permet des traitements d'information d'un genre tout nouveau.

On a cru tenir des mémoires de ce genre vers 1961, lorsqu'on songea à utiliser le phénomène dit de supraconductivité. Si dans un métal porté à la température très très basse de l'hélium liquide, c'est-à-dire à —270 °C (à 3° du zéro absolu), vous lâchez une fois du courant électrique, il continuera d'y circuler éternellement, tant du moins que la température ne remonte pas, ce qui veut dire que la résistance électrique du métal devient nulle à cette température. Elle peut toutefois revenir à une valeur non nulle si on établit autour du circuit un champ magnétique suffisant. Cela permet de provoquer à volonté la persistance ou la disparition de « courants éternels » en divers points. A partir de là, et par des astuces incroyables dont je n'ai pas le temps de vous parler, on arrivait à créer des mémoires *associatives*. Malheureusement cette technique n'a pas exaucé jusqu'ici les espoirs qu'elle avait fait naître, d'abord parce que le basculement de l'état supraconducteur à l'état résistif n'est pas plus rapide, est plutôt plus lent, que le basculement magnétique des tores de ferrite. Ensuite, parce que tout le calculateur doit passer toute sa vie plongé dans l'hélium liquide, et qu'on n'a toujours

pas trouvé le moyen de préparer et entretenir ce précieux produit autrement que par une véritable petite usine. Pour l'heure, le problème de créer une grande mémoire statique ultrarapide et *associative* reste posé sans solution convaincante.

Ne croyez pas que la recherche de nouveautés ne se porte que sur la logique et la mémoire. Les périphériques aussi connaissent la fermentation intellectuelle et technique incessante et prodigieuse qui a marqué le monde *ordino* dès son berceau.

On trouve de plus en plus en usage des machines qui savent « lire » directement du texte écrit, du moins quand il est typographié ou tapé à la machine : c'est-à-dire qu'on présente, sous une tête faite, comme notre œil, d'une mosaïque de cellules photoélectriques, une page de texte, et en quelques secondes toutes les lettres de tous les mots de toutes les lignes ont été reconnues et transformées dans leurs codes top-tip, qui peuvent entrer, soit directement dans la mémoire centrale si celle-ci a assez de place, soit dans une mémoire séquentielle (bande, disques ou tambour). D'autre part on développe aussi des matériels qui permettent d'impressionner directement des bandes magnétiques en tapant un texte en clair sur un clavier et visualisant en même temps le remplissage de la bande par les tops-tips engendrés.

Côté sortie, l'imprimante à tambour, que j'admire tant, est déjà talonnée par deux techniques concurrentes : d'une part, la reproduction, soit au polaroïd, soit en xérographie, d'un texte qu'on fait apparaître sur la face d'un tube cathodique, ce qui permet en particulier d'imprimer une typographie plus variée; d'autre part, des jets extra-fins d'encre à séchage instantané, jets constitués par des gouttelettes électrisées et que de ce fait on peut dévier à volonté entre deux condensateurs recevant des signaux électriques, un peu comme on dévie le faisceau d'électrons par les deux bobines déviatrices d'un tube cathodique.

La recherche sur les interfaces, ainsi que sur tout le hardware, ne va pas seulement dans le sens du perfectionnisme et du gigantisme, mais aussi dans celui de la simplification. Toute une série de laboratoires ont parié sur le petit calculateur des familles, pour ne pas dire « de poche ». La mémoire séquentielle de ces vrais futurs « Puck » sera un simple magnétophone à minicassettes, comme ceux que vous achetez pour 400 F. Mais ne vous ai-je pas conté moi-même que le déroulement des bandes magnétiques dans ces gadgets était trop incertain, mécaniquement, pour pouvoir servir de base à un adressage? C'était vrai. Mais si on n'est pas trop exigeant sur la vitesse, on peut s'en tirer en plaçant dans chaque petit bloc qui défile un petit compartiment spécial donnant l'identité et l'adresse du bloc par un code, éventuellement une manière d'inscrire des traits et points de type Morse sur une piste. Le reste de l'ordinateur reconnaît alors au vol, au moyen d'un petit programme d'interprétation, de quel bloc il s'agit. On peut dire qu'on a échangé une complication de hardware contre une complication de software. Au détriment certes de la vitesse, mais celle-ci a atteint une valeur suffisante pour des applications au foyer, comme de tenir les comptes du ménage, sortir des adresses de fournisseurs, voire aider les travailleurs de la famille dans leurs emplois. On voit déjà sortir ces mémoires à minicassettes. Quant aux unités de contrôle et de calcul, si on les réalise tout entières en technique « MOS », et avec des performances simplifiées, on pourra les faire tenir dans le volume d'une grosse machine à écrire. Et on étudie très sérieusement une petite boîte qui, adjointe à votre récepteur TV, peut le faire servir de console de visualisation...

Quant au software, il continuera à dérouler devant nos yeux des langages de toutes sortes, des programmes préfabriqués en boîte de plus en plus ambitieux,

des programmes de service, etc. La plus importante nouveauté dans ce sens sera sans doute l'extension de diverses formes de multiprogrammation dont nous parlerons dans la troisième partie.

Avant de clore ce chapitre, je vous demande la permission (une fois de plus) de laisser paraître mon admiration. J'espère vous l'avoir fait partager en déroulant devant vous le panorama des merveilles d'intelligence, d'imagination et de ténacité, doublées de fabuleuses réussites technologiques, qui ont permis de réaliser l'ordinateur d'aujourd'hui et qui continuent à fleurir et foisonner à un rythme fantastique pour préparer l'ordinateur de demain. De 1955 à 1975, on aura sans doute multiplié sa puissance par cinq tous les cinq ans. Jamais l'humanité ne s'était orchestré un tel festival. La capacité, la vitesse, la compacité, la fiabilité ont augmenté en 20 ans, dans le monde de l'ordinateur, dix à cent fois plus que dans le monde de l'aviation en 70 ans. Pendant que le prix baissait considérablement au lieu d'augmenter.

Tous les domaines qui se trouvent dans le sillage de l'ordinateur, la mécanique, l'électronique, le magnétisme, l'optique, la linguistique, la logique, sont soulevés vers leurs sommets par sa fulgurante avance, ou portés à leurs limites. Je n'arrive pas à comprendre que certains de nos contemporains restent insensibles devant ces exploits. Que dis-je, beaucoup les trouvent déplaisants et se voilent la face.

▶▶ — Ces critiques ne nient pas la grandeur, et même si vous y tenez, la beauté des exploits techniques et scientifiques accomplis pour réaliser les ordinateurs; ils regimbent devant les applications qu'on en fait.

▶ — Ces applications seraient donc maléfiques? Voyons cela.

Vous avez parcouru les étapes suivantes de la courte mais prodigieuse histoire des calculateurs.

Fin de la guerre	: ENIAC, le premier gros calculateur à lampes; un diplodocus.
1946-48	: les idées de von Neumann pour programmer.
1948-52	: progrès lents dus à l'insuffisante fiabilité, premières mémoires à ferrites.
1952-54	: premières machines tout parallèle, premiers lecteurs de bandes magnétiques.
1954-58	: maturation simultanée du transistor, du tore de ferrite et des langages de programmation.
1958-65	: développement tumultueux de machines de plus en plus puissantes et sûres, du disque magnétique, des grandes imprimantes.
1965-70	: apparition des circuits intégrés, commercialisation du software, apparition de la console de visualisation et du mode conversationnel.
Pendant tout ce temps	: baisse des prix de 10 fois et amélioration des performances de 100 fois.

Après ce voyage dans le passé vous avez eu droit à quelques visions d'avenir : les imprimantes à flux d'encre dirigé, la mémoire optique, les calculateurs de poche, le temps partagé, le télétraitement, etc.

Troisième partie

les applications

La gestion

La documentation

Les applications « en temps réel »

Médecine et enseignement

Applications scientifiques

L'économie politique des ordinateurs

9

La gestion

Les calculateurs électroniques sont nés pour effectuer les calculs les plus complexes demandés par la solution de problèmes scientifiques et techniques, calculs infaisables à la main. Pendant des années on ne leur vit pas d'autre finalité. Puis, à mesure qu'on reconnut plus clairement qu'ils traitaient en fait toutes les données codables, selon des processus qui ne se limitaient pas aux opérations arithmétiques, on s'enhardit à envisager des applications à la vie des affaires. Dans les années 1956-62, ce tournant, qui parut audacieux et redoutable, occupa tous les esprits. Est-ce vraiment possible? Est-ce rentable? N'impose-t-on pas une tyrannie mécanique aux bureaux? Les premières tentatives furent, ainsi qu'il était naturel, encombrées d'innombrables erreurs humaines souvent difficiles à dénicher et assombries par les pannes des machines encore fréquentes à cette époque. Les sceptiques avaient la partie belle contre les optimistes tenaces (qu'on appelait derrière le rideau les obsédés), de sorte que la « recherche d'âme » sur l'emploi de l'ordinateur à la gestion dura des années. C'est avec le sourire qu'on regarde aujourd'hui ces doutes et ces tribulations. L'ordinateur s'est en fin de compte tellement imposé dans la gestion, que le grand public finit par croire qu'il n'est quasiment fait que pour ça. Le mot « **informatique** » est même souvent entendu comme recouvrant seulement les activités de gestion assurées par le moyen, ou avec le secours, de l'ordinateur.

Le triomphe de celui-ci dans les activités de gestion résulta d'abord de l'amélioration considérable des matériels et de l'art de la programmation, qui délivra les usagers du cauchemar d'interminables mécomptes. Ensuite du fait qu'on dégagea le noyau unificateur de tout ce domaine : *le traitement des fichiers*. En fin de compte, et vous allez le vérifier, la grande majorité des problèmes qu'on rencontre dans ce secteur se réduit à *gérer des fichiers* : sortir des données sélectivement, introduire des rubriques, unifier ou séparer des listes, mettre à jour un fichier par l'introduction d'informations nouvelles ou la modification ou l'annulation d'informations anciennes, copier un fichier d'un support sur un autre, etc. Ces tâches font appel surtout à des opérations *logiques* — coïncidence, mélange, complémentation, tri, comptage — et peu à des opérations *arithmétiques* (addition et multiplication). Elles ont été étudiées pour elles-mêmes avec toute la patience et l'habileté de la méthode scientifique, ce qui a permis d'édifier un véritable *corps de doctrine*, lequel à son tour a permis d'écrire des programmes *universels*, préfabriqués, pour l'exécution de toutes ces tâches quelle que soit la nature des informations fichées. Cette science de la manipulation des fichiers, dotée de langages et d'outils raffinés, a contribué puissamment à l'entrée en force du calculateur dans les problèmes de gestion.

▶▶ — Quels sont ces problèmes ?

▶ — Payer les employés, tenir les comptes en banque ou les polices d'assurances, gérer des stocks, réserver des places dans les avions, etc. Je ne vous parlerai pas des problèmes de paie ou de banque, car vous savez déjà assez d'informatique pour comprendre qu'un ordinateur peut établir la feuille de paie du personnel, par exemple si on lui fournit toutes les données codées sur le travail et la personnalité des employés ainsi que les barèmes de salaires. De même pour les comptes en banque, s'il connaît les dépôts, retraits et ordres d'achat. Je ne vous donnerai quelques détails que sur deux problèmes de gestion qui soulèvent des problèmes originaux.

Avant de les développer, je voudrais toutefois vous dire que la puissance de l'ordinateur en matière de gestion est telle qu'on a tendance à en *abuser*. Par exemple, on multiplie à l'envi les classes, sous-classes, cas, sous-cas dans le barème des salaires, en se disant qu'il suffira à l'analyste et au programmeur d'insérer quelques instructions supplémentaires dans le programme de l'ordinateur pour que celui-ci en tienne compte et établisse la feuille de paie en conséquence aussi vite, pratiquement, que sans ces multiples subdivisions. On en fait autant dans les domaines plus importants, d'intérêt national, comme la Sécurité sociale qui est devenue une forêt touffue de nombreux milliers de cas. L'ordinateur, pense-t-on, s'y reconnaîtra. Sans doute. Mais le citoyen ? Le patron ? Et le législateur même ? Eux s'y perdront. Alors qu'il n'y a pas de démocratie, pas d'humanisme possible si les rapports juridiques entre les individus ne sont pas assez simples, assez transparents pour que l'esprit de chacun puisse les appréhender et s'y mouvoir à l'aise. Sur ce plan l'ordinateur apporte, pourrait-on dire, un *excès* de services. Il suscite un peu chez les organisateurs des affaires cette paresse mentale que la télévision répand parmi les consommateurs de loisirs. C'est une pente à laquelle il importera de résister.

La tenue des stocks

Je vous ai déjà mentionné cette activité en passant. Elle est devenue l'un des domaines privilégiés de l'informatique de gestion, et elle est intéressante parce qu'elle soulève des aspects nouveaux.

Je veux garnir les rayons de mes multiples succursales de façon optimum, c'est-à-dire avec assez d'articles pour être sûr que je pourrai satisfaire toute demande, mais pas trop pour que ma charge financière ne devienne pas prohibitive. Comment déterminer cet optimum ?

Je suppose qu'une série de dispositions administratives astucieuses permettent à l'ordinateur de connaître l'état du stock à la date J d'aujourd'hui pour tout article A dans toute région R (j'aurai dû pour cela imposer à mes magasiniers de tenir leurs états bien à jour et de les communiquer par clavier tous les soirs à l'ordinateur). Il faut maintenant que je compare la quantité Q de A que j'ai en magasin le jour J dans la région R avec une quantité idéale Q_i qui constituerait l'optimum. Je commanderai un supplément de A si je trouve Q inférieur à Q_i, et je transférerai une partie de A de la région R vers une autre si je trouve Q supérieur à Q_i.

▶▶ — Mais d'où vais-je tirer cette quantité idéale Q_i ?

▶ — Bien entendu, de statistiques antérieurement accumulées sur la demande

de l'article A dans la région R à l'époque J (demande qui peut dépendre de la région si l'article A est lié au climat ou à l'activité locale dominante, et de l'époque si l'article est saisonnier ou lié à la mode). Cette statistique est très difficile à tenir et délicate à interpréter à cause de l'incidence qu'ont sur les ventes les facteurs mêmes qu'on veut leur adapter, tels que la qualité du service commercial. Les indices que livre cette statistique sont donc souvent *dépendants*, et obligent d'écrire des *équations*, où les variables cherchées jouent un rôle complexe à la fois de cause et d'effet. En outre la tenue même des statistiques, je veux dire leur tenue matérielle, exige que des flots d'informations soient codés, transmis et enregistrés tous les jours. Sur ce point l'ordinateur qui calcule chaque jour la composition idéale du stock doit nécessairement se servir d'un autre ordinateur qui emmagasine les statistiques des ventes antérieures et lui en livre des extraits sur commande.

Mais le plus grave n'est pas encore là. Le plus grave, c'est que même les statistiques les plus complètes sur les ventes anciennes ne livrent jamais de certitudes, au plus des probabilités. Car les ventes sont soumises à des fluctuations en partie imprévisibles, soit parce qu'intrinséquement aléatoires, soit parce que dépendant de facteurs impossibles à cerner tels que la situation économique générale ou l'apparition d'articles concurrents. On ne peut donc pas demander à l'ordinateur une valeur exacte de la quantité idéale Q_i, qui n'existe pas. Au mieux peut-on faire calculer à l'ordinateur ce qu'on appelle la « courbe de distribution » des quantités de l'article A vendues dans la région R à la date J, c'est-à-dire une courbe qui, pour chaque valeur possible de la quantité de vente, dit combien de fois on l'a effectivement observée. Si cette courbe présente un maximum prononcé pour une certaine valeur de la quantité, cette valeur est « la plus probable » et constitue la meilleure candidate possible pour Q_i. Si on trouve des courbes plus tarabiscotées, le seul recours est de faire « tirer » par l'ordinateur une valeur de Q_i au hasard parmi des valeurs qu'on lui propose, affectées de poids conformes aux fréquences avec lesquelles elles se présentent dans la courbe de distribution.

Cette opération s'appelle « tirage d'une valeur aléatoire dans une loi de probabilité ». Elle exige un sous-programme spécial assez complexe, qui a ceci d'étonnant qu'il organise l'exécution d'un acte sortant volontairement d'une règle. Par exemple, supposez que la quantité étudiée présente 100 valeurs possibles, réparties dans 10 000 cases d'une mémoire, une valeur prise souvent d'après la loi de probabilité étant répétée dans plus de cases qu'une valeur prise rarement, en proportion des fréquences relatives assignées par la loi. Si j'appelle n'importe laquelle des 10 000 cases au hasard, c'est-à-dire avec la même chance de 1/10 000 pour chacune, la valeur qui me sera débitée constituera bien un tirage aléatoire respectant la loi de probabilité. Pour faire cet appel, le programmeur peut avoir inscrit n'importe quelle adresse entre 0 et 9 999, à sa fantaisie, ou l'adresse lue sur des roulettes qu'il stoppe au hasard. Vous voyez qu'il faut consacrer une mémoire spéciale à être remplie avec des valeurs reproduisant la statistique observée, ou un bloc spécial d'une mémoire de masse. En réalité les solutions retenues sont un peu différentes de celle que je vous ai décrite, mais je ne m'attarderai pas sur les détails techniques, je voulais seulement vous montrer, à l'occasion de ce problème du remplissage optimum des stocks, combien de choses curieuses on doit pouvoir programmer sur les ordinateurs.

Le reste de la gestion des stocks est plus routinier. Une fois déterminé l'approvisionnement à faire, l'ordinateur en notifie par un texte les chefs magasiniers, ou

encore fait rédiger lui-même, par un programme d'édition, les bons de commande si les fournisseurs figurent tous en mémoire avec leurs adresses et catalogues. A l'autre extrémité, l'ordinateur absorbe tous les bons relatant les ventes faites ou les commandes reçues, bons qui ont dû être conçus de manière à être directement lisibles par la machine. On peut aussi faire faire à l'ordinateur des *plans de livraison*. A cet effet, il faut qu'un analyste ait préalablement assigné une note à chaque parcours de camion à travers la ville ou la région donnée, note illustrant la facilité du parcours en fonction des endroits traversés et de l'heure. Il faut ensuite écrire un programme très savant pour résoudre le problème de trouver un *cheminement optimum* entre les lieux où il faut livrer. Si les magasins sont équipés en science-fiction, les paquets désignés par l'ordinateur sortiront tout seuls de leurs cellules par des tapis roulants qui les enverront directement vers le ventre des camions dans l'ordre inverse des points d'arrêt à visiter...

La réservation des places d'avion

Jadis, quand vous alliez demander votre Paris-Chicago au guichet d'une compagnie aérienne pour tel jour, l'aimable demoiselle enregistrait vos coordonnées et le lendemain vous téléphonait pour confirmer ou infirmer s'il y avait une place pour vous. S'il n'y en avait pas, une seconde négociation commençait pour savoir si vous pourriez passer par Montréal, ou voyager un autre jour.

C'était dans les temps heureux (ou malheureux?) où votre agenda présentait de nombreux blancs, tandis que 200 vols se partageaient tous les jours l'espace aérien du globe. Aujourd'hui il y en a 6 000, votre planning de travail et voyages est devenu trépidant, et la clientèle s'est multipliée par 10 ou 20. L'ancien système est donc devenu impossible et la navigation aérienne commerciale aurait dû s'arrêter par l'engorgement des guichets si l'ordinateur n'était venu prendre la relève. Grâce à quoi vous allez toujours au guichet, et l'aimable demoiselle prend toujours note de votre demande, mais elle la tape sur un clavier qui la transmet à un ordinateur, et 10 secondes plus tard, au plus, arrive la réponse, et quelle réponse! Non seulement oui, place libre mais encore, si c'est non, proposition de plusieurs variantes. Si vous dites oui pour le vol que vous aviez demandé ou pour l'une des variantes offertes, vous déclinez vos coordonnées, la demoiselle les rentre à nouveau par son clavier dans l'ordinateur, et la réservation est faite.

Et ce n'est pas tout. La machine vous dit quel temps il fera à votre atterrissage, et quel film on projettera pendant le vol. Un de ces jours, elle vous donnera le nom, l'âge, la profession de votre voisin et, si c'est une voisine, pourquoi pas, ses mensurations...

Ce service automatisé, devenu indispensable, met en œuvre des moyens d'informatique considérables. Songeons un peu ensemble à tout ce qu'il faut avoir enregistré et prévu.

Il faut avoir enregistré dans une mémoire tous les vols du monde, chacun ayant reçu en code un numéro d'identification suivi d'une étiquette complète comprenant ses escales, ses horaires, le type d'accommodation qu'il offre. Il faut en outre, si on veut chercher des variantes au cas où le vol direct demandé est plein, inscrire pour

chaque aéroport des indices donnant sa distance et son temps d'accès par rapport à une série d'aéroports « contigus », afin qu'un programme consultant tous les parcours « voisins » possibles trouve des critères permettant de déterminer les plus favorables.

Il faut encore que l'état d'occupation de tous les vols sur plusieurs mois à l'avance soit tenu à jour perpétuellement, dans une immense « table » que le programme fait consulter à la seconde. Cette « table » contient à chaque instant le nombre des places restant libres dans chaque vol, et ce nombre diminue de x chaque fois que x réservations viennent d'être faites sur ce vol, jusqu'à ce qu'il vaille 0, auquel cas la table répond « complet » quand on la consulte. Et cette réponse déclenche les pas du programme qui ordonnent la consultation des variantes, avec tous les ordres conditionnels que vous pouvez imaginer : si le vol suivant de la même ligne est distant de plus de quatre heures, voir les vols dont l'indice de proximité donne une arrivée dans un délai plus court, mais les écarter s'ils supposent une étape de nuit hors de l'avion, considérer le supplément de prix, s'il est trop grand chercher des vols moins voisins dans le temps et l'espace mais moins coûteux, pour chaque itinéraire de remplacement acceptable selon ces critères examiner l'état d'occupation, etc. Cette recherche par programme doit d'ailleurs pouvoir se combiner avec des initiatives émanant du guichet.

Et ce n'est pas tout. Une fois un accord établi avec le candidat voyageur, il faut que ses coordonnées à lui rentrent dans une autre mémoire, celle des passagers, qui soit corrélable avec la table des vols par des indices de connexion, de façon qu'on sache à tout moment non seulement combien de places sont retenues sur chaque vol, mais encore qui occupe ces places, car on doit pouvoir s'entretenir avec les occupants en cas de retard, annulation, etc. Et ce n'est pas tout. Les passagers sont des êtres fantasques, qui changent souvent d'avis ou d'état de santé, de sorte qu'il serait imprudent de fixer les disponibilités d'après les seules réservations, faites parfois des semaines avant le vol effectif. Il faut laisser flotter autour des nombres de places, des marges qui elles-mêmes sont déduites des statistiques de fluctuations qui s'observent aux diverses époques sur les diverses lignes. Ces marges sont évaluées par une section de l'ordinateur qui calcule sans être consultée, parallèle avec le système de réservations, et corrige en permanence, silencieusement, la table d'occupation. Et ce n'est pas tout. Une fois la réservation faite, il faut aller consulter une autre mémoire où s'inscrivent les prévisions météo et les autres caractéristiques du vol qu'on doit communiquer aux passagers.

Vous pensez bien que la conception d'un système de cette envergure exige des années de travail d'énormes équipes d'analystes et programmeurs. Chez Air France, le programme de consultation comprend 90 000 pas d'instructions écrites en langage FORTRAN, ce qui correspond à 400 000 ordres machine. Le programme de mise à jour et de contrôle des fichiers en comprend 50 000. Il faut 1 000 instructions rien que pour assigner des tours de service aux divers guichets qui posent des questions en même temps, compte tenu de leur importance commerciale et de la disponibilité de l'ordinateur à ce moment. Quant au hardware capable de traiter en quelques secondes un tel nombre de questions aussi complexes, il se situe lui aussi à la limite des performances actuelles. La table d'occupation des vols fait plus de 50 millions de caractères et la mémoire des dossiers des passages en fait 150 millions (une grande compagnie a en permanence 500 000 à 1 000 000 de dossiers en mémoire). Comme il faut pouvoir y accéder dans des temps de l'ordre de la milliseconde, la bande magnétique et la

pile de disques magnétiques sont toutes deux exclues, seul reste possible le tambour magnétique, dont c'est ici le royaume. Quant aux guichets, comme ils doivent pouvoir correspondre avec l'ordinateur en mode conversationnel, ils ont fourni le point d'application rêvé pour les consoles à tubes cathodiques. Celles-ci doivent en fait leur développement rapide à la réservation automatique des places sur les lignes aériennes.

Mais le plus grave à ce sujet, je l'ai gardé pour la fin. C'est que les postes qui conversent entre eux sont répandus sur tous les coins du globe. Le guichet de Tokyo est aussi souvent intéressé à une réservation Francfort-Johannesburg que celui de Paris à une réservation Tokyo-Bangkok ou Washington-Mexico. Il faut donc que tous les guichets puissent envoyer leurs questions à l'ordinateur, et recevoir de celui-ci des réponses, en messages codés sur des lignes télégraphiques faisant le tour du monde.

Que les informations puissent circuler entre un guichet et un ordinateur, ou éventuellement entre deux ordinateurs, par des fils ou des ondes téléphoniques ou télégraphiques, voilà qui ne devrait présenter pour vous, cher lecteur, aucune difficulté conceptuelle. Je ne comprends pas, pour ma part, les mystères qu'on fait autour de ce qu'on appelle le « **télé-traitement** ». Puisque vous avez compris que les signaux qui travaillent dans les ordinateurs sont des tops et tips électriques, et qu'on peut faire entrer ceux-ci dans ceux-là au moyen d'organes périphériques appelés aussi des « interfaces », il n'y a aucune difficulté à comprendre que les tops et tips puissent être envoyés à une interface, par fil ou par radio, depuis des postes lointains. Le seul problème qui émerge alors, mais au premier plan, c'est celui des erreurs. Car au cours du voyage des signaux sur de longs circuits à travers les terres, les mers et les airs, on n'est plus maître des signaux en chaque point comme on l'est dans le sein d'un ordinateur. Or, alors qu'en téléphonie ou télégraphie de parole quelques glissements ne sont pas très graves, parce que l'homme restitue facilement le sens, il n'en est pas de même lorsqu'on transmet à distance sur des circuits, des *données* destinées à un ordinateur. Toujours à cause de la bêtise de ce dernier, s'il reçoit le code de Munich au lieu de celui de Johannesburg, il renverra à Tokyo une réponse sur un vol Francfort-Munich au lieu d'un vol Francfort-Johannesburg, et Tokyo n'aura aucune raison de suspecter la méprise. Voilà une erreur indétectable qui peut engendrer de jolis drames. Il faut donc que la circulation des données sur les circuits de télécommunication ait une très grande sécurité, et cette exigence drastique est d'autant plus sérieuse qu'elle va à l'encontre d'une exigence de vitesse. La technique des télécommunications est en effet telle que plus on veut tasser de signaux dans la seconde, plus le risque d'erreurs augmente. Sur des lignes ordinaires on arrive difficilement, au prix de soins spéciaux donnés aux équipements des centrales, à passer sans erreur plus de 2 500 tops ou tips par seconde. Pour aller au-delà, il faut créer des codes spéciaux extraordinairement savants, qui se redressent eux-mêmes (on les appelle « autocorrecteurs »), ou employer des *lignes spéciales*. C'est pourquoi ce qu'on appelle la « transmission de données » est devenue une discipline en soi, et un défi majeur aux réseaux de télécommunications. En Amérique ce défi a été relevé frontalement, et dans quelques années ce pays sera doté d'un formidable réseau de lignes spéciales pour transmissions de données dont la capacité de transmission sûre sera 100 fois celle du réseau téléphonique et télégraphique actuel, cependant très dense et qui n'a été porté à son développement actuel qu'en 50 ans. C'est un exemple de plus du phénoménal dynamisme que met en branle l'apparition de l'ordinateur. En France, on est encore très loin de compte,

mais l'équipement du territoire pour la transmission de données avance plus vite aujourd'hui que la finition de son équipement téléphonique.

Dans tous les cas, la transmission à distance de données binaires fait appel à au moins une méthode universelle pour la détection des erreurs, c'est celle qu'on appelle « le digit de parité ». Supposez que la « tranche » d'information qu'on transmet d'un seul coup soit un caractère, ou un mot, ou une fraction de mot occupant, mettons, 7 digits. Au moment où je l'émets, je sais ce que la tranche signifie et quel tableau de tops et tips elle comprend. Dès lors, je peux ouvrir une huitième place, non porteuse de « sens » sur l'information, et y loger un digit artificieux calculé chaque fois de telle manière que la somme des digits du tableau *avec celui-là* soit, mettons, ***paire.*** Puis, au détour de chaque passe dangereuse dans la transmission, j'installe un petit additionneur qui fait la somme de tous les digits. S'il trouve une valeur impaire, il stoppe la transmission et alerte les surveillants, qui peuvent facilement repérer l'endroit fautif, puisqu'il sera certainement compris entre l'additionneur qui a crié gare et le précédent qui n'a rien dit. Ce système simple et joli n'apporte naturellement pas 100 % de garantie, car il pourrait se faire que l'erreur frappe deux digits à la fois de façon que la somme du tableau garde la même parité. Mais une telle coïncidence est extrêmement rare dans un tronçon compris entre deux stations de contrôle. On utilise la méthode aussi dans le sein même des ordinateurs, mais là on demande non seulement que les erreurs soient signalées, il faut encore qu'elles se corrigent automatiquement. C'est une autre histoire, dans les méandres de laquelle vous me dispenserez d'entrer.

L'on a vu que pour gérer des stocks, il faut inventer des règles permettant de tirer des suites de nombres se suivant sans règle. Et l'on a apprécié, à propos du problème de réserver des places dans les avions, l'énormité que peut prendre un programme d'ordinateur.

10

La documentation

L'homme étouffant sous son savoir?

Vous avez tous entendu dire que notre civilisation est menacée d'étouffement parce qu'il est impossible aux spécialistes de se tenir au courant de tout ce qui se pense, se découvre, s'invente et se publie dans leurs domaines. Est-ce vrai ou non? La question est complexe et ce n'est pas le lieu de l'élucider ici. Sachez seulement, pour vous tranquilliser un peu à son sujet, que la complainte sur l'excès des connaissances qui saturerait les chercheurs ne date pas d'aujourd'hui. Des cris d'alarme furent poussés dès 1620, eh oui, sur la grave crise d'indigestion qui menaçait le progrès des sciences parce qu'il était inconcevable qu'un maître pût ingérer le torrent de ce qui se publiait à l'époque, et qui se montait, disait le plaignant, à « près de 60 ouvrages par an »! Songez donc! Quel esprit rassis peut lire 60 ouvrages par an?

S'il n'est pas évident que notre civilisation va connaître une horrible mort par satiété, on ne peut nier non plus que l'engorgement des travaux (comme, sur un autre plan, l'engorgement des biens) lui pose un problème très sérieux. Voici en effet quelques chiffres illustrant l'énormité de la production annuelle de travaux intellectuels dans le monde :

- 250 000 titres d'ouvrages dont 40 000 savants;
- 120 000 revues publiant chacune 8 articles par numéro à raison de 8 numéros en moyenne par an;
- 100 000 thèses de doctorat;
- des millions — impossibles à compter — d'articles de presse et de documents plus ou moins sauvages, notes volantes, prospectus, fiches, bulletins d'innombrables laboratoires et agences, rapports d'études commandées;
- sans compter les documents non textuels ou non imprimés, images, photos, dessins, cartes, discours, disques, etc.

Il faut bien situer ce problème si on veut comprendre comment l'ordinateur peut aider à le résoudre.

Traitons-le sur un exemple.

Supposons qu'un chercheur s'intéresse à la prédilection des moustiques pour les blonds (et blondes). Comme tous les sujets concevables, fussent-ils très spécialisés et apparemment futiles, celui-ci a été traité dans une bonne cinquantaine d'articles, notes et passages de livres qu'on appelle des « références bibliographiques ». Et c'est

un cas relativement modeste. Bien souvent la « littérature » concernant un sujet contient des centaines ou milliers de références. Le problème est de détecter cette littérature dans l'Himalaya de la production littéraire.

On a eu pendant longtemps l'espoir d'y arriver en classant la production littéraire (l'adjectif « littéraire » ici ne s'oppose pas à « scientifique » mais désigne simplement tout ce qui a été pensé, composé, écrit par un auteur dans quelque discipline que ce soit) comme on classe les animaux, par genres, embranchements, familles, espèces, etc. Ainsi notre sujet appartiendrait au genre science, embranchement zoologie, famille insectes, espèce moustique. etc. Si on range toute la production dans des boîtes, sous-boîtes et sous-sous-boîtes selon cette classification hiérarchisée, on peut espérer trouver dans une sous-sous-sous-boîte déterminée tout ce qui a été écrit sur les moustiques. Mais outre que la recherche ne serait pas facile à faire par ordinateur, il resterait que les mœurs alimentaires ne constituent qu'un parmi les multiples points de vue sous lesquels les savants ont étudié les moustiques, les autres étant, par exemple, le bruit qu'ils émettent, leurs pattes, leur reproduction, leur habitat, etc. Et même du point de vue des mœurs alimentaires, il y a fort à parier que beaucoup d'études traiteront de la cadence, de la chimie, de la quantité des nourritures moustiquaires, et il faudra une nouvelle recherche pour découvrir dans cette masse les études qui s'intéressent à la corrélation entre ces nourritures et la complexion cutanée ou chevelue des victimes.

Les mots-clés

On s'est vite aperçu que la variation infinie des *points de vue* ne permettait pas d'enserrer tous les sujets d'étude du monde dans une classification *unique dressée a priori.* On doit renoncer à saisir un travail d'après le *cadre logique* dans lequel il s'insère. Le seul espoir est de décrire en abrégé *son contenu même*, sans se demander dans quelle niche de la science humaine il se loge. Pour cela, un analyste, au courant du grand domaine dans lequel se situe le travail, le lit (en diagonale) et dégage ses **« mots-clés »,** dont l'ensemble exprime au mieux de quoi il traite. Par exemple, un article concernant le sujet qui nous intéresse aura été « indexé » par l'analyste avec les *mots-clés* suivants : « moustique, nourriture, habitudes, blonds ». Vous observerez la puissance du procédé : la simple adjonction de « blonds » comme mot-clé suffit à écarter la masse des études qui concernent la nourriture des moustiques où on ne se préoccupe pas de leur corrélation avec les blonds.

Faisons maintenant un grand bond et supposons que des légions d'analystes ont ainsi indexé tous les travaux parus dans le monde avec des *mots-clés* caractérisant leur contenu. En moyenne, il en faut une dizaine pour bien situer un document (parfois davantage, songez par exemple à une étude juridique sur l'héritage par les collatéraux de biens ruraux achetés par l'un des époux lors d'un premier mariage sous le régime de la communauté). Bien entendu nous aurons quand même une classification pour subdiviser *un peu* et ordonner *au premier degré* l'Himalaya documentaire, ne serait-ce que parce qu'il faut spécialiser les analystes, qui ne peuvent pas indexer proprement les documents s'ils n'entendent rien à la matière traitée. On aura donc, mettons, 200 *grands domaines* dans lesquels, d'ailleurs, la littérature s'articule d'elle-même, domaines du genre : électronique, océanographie, biologie génétique,

sociologie des sociétés primitives, immunologie, insectes, théâtre, psychanalyse, droit commercial, etc. Dans l'un de ces domaines, des analystes « indexent » tous les textes parus en leur donnant : — d'une part, un « *label* » qui repère et identifie le document *indépendamment de son contenu*; — d'autre part, une « *étiquette* » qui décrit son contenu, *indépendamment du label*, par les mots-clés qu'on y rencontre.

Ne croyez pas qu'aucune de ces deux activités soit facile. Dans le « *label* » doivent en effet figurer des quantités de choses : le titre, le ou les auteurs, l'éditeur, la date, la pagination, le format, la langue, la revue ou l'ouvrage d'où le document est extrait, la commande d'étude à laquelle il répond (si le texte est un rapport rédigé pour une étude commandée) et bien d'autres éléments de « signalisation ». Ceux-ci deviennent vite si nombreux et touffus qu'il faut, pour les manier proprement, créer des normes remplissant des volumes.

Quant à l' « *étiquette* » comprenant les mots-clés d'un texte, elle pose plus d'un piège. D'abord les allitérations. Dans un texte qui concerne la gestion des malades hospitalisés, dois-je écrire « hôpitaux » ou « hôpital »? Si plus tard j'interroge l'ordinateur avec le premier mot, il ne me produira pas les études où figurait le second. Après les allitérations viennent les synonymes. Dois-je prendre pour mot-clé hôpital, ou clinique, ou maison de santé? Et dans ce dernier cas, s'agit-il de deux mots-clés, maison et santé, ou d'un seul? Et pour le présent livre mettrai-je comme mot-clé seulement « informatique », ou encore « traitement de données », « ordinateur », « calculatrices électroniques »? Puis viennent les problèmes plus graves de syntaxe. Si je demande à l'ordinateur de me sortir les textes qui contiennent les mots-clés « acier » et « outils », donnés comme ça tout nus et inconnexes, il me sortira aussi bien les études concernant les outils pour travailler l'acier que les aciers pour faire des outils, alors que je ne voulais que les unes et pas les autres. Si je souhaite qu'on m'extraie tous les travaux concernant l'influence du protestantisme sur les penseurs libéraux du XVIII^e siècle, et que je ne donne que les mots-clés isolés « influence », « protestantisme », « penseurs libéraux » et « XVIII^e siècle », sans liens syntaxiques entre eux, ce jeu de billes peut correspondre à une vingtaine de thèmes entièrement différents, tels que « l'influence du XVIII^e sur le protestantisme des penseurs libéraux », ou « le libéralisme du XVIII^e a-t-il influencé les penseurs protestants », etc.

Vous voyez que le principe, au demeurant excellent, des mots-clés, appelle encore tout un cortège de raffinements avant de pouvoir servir efficacement : une standardisation des orthographes, des dictionnaires pour renvoyer à partir de toutes les désignations possibles d'un concept vers l'unique qui a été retenue, et surtout des signes ou mots supplémentaires pour préciser quelles connexions logiques règnent entre les mots-clés (cause à effet, particularisation ou généralisation, antécédence ou conséquence logique, etc.). Lorsque les mots-clés avec lesquels on décrit le contenu d'un document ont été ainsi purifiés, standardisés, isolés, et dotés d'une petite syntaxe, on les appelle des « ***descripteurs*** ». L'ensemble des *descripteurs* dont on se sert pour « indexer » les documents d'un domaine, ainsi que des symboles qui les unissent, s'appelle un « ***thesaurus*** ». Le thesaurus d'une discipline telle que l'électronique ou l'immunologie utilise environ 8 000 descripteurs. Quant à l'ensemble des fiches portant d'une part les *labels* des documents conservés dans un centre documentaire et, d'autre part, leurs *étiquettes* descriptives écrites avec le *thesaurus* convenu, il s'appelle le « ***corpus*** ». On est donc parvenu à la science, puisqu'on a donné aux choses dont on parle une forme incompréhensible aux profanes, et des titres latins.

Vous venez demander une bibliographie

Imaginons maintenant une bibliothèque idéale dans l'un de nos 200 grands domaines, bibliothèque où trône le corpus complet de toutes les indexations, de tous les travaux parus jusqu'à ce jour dans ce domaine, et bien entendu aussi ces travaux mêmes. A mesure que des documents nouveaux y entrent, des analystes établissent leurs fiches, des photographes les microfilment, et des appariteurs rangent tout ça sur des rayons ou dans des boîtes. D'autre part, des informaticiens rentrent tous les contenus de toutes les fiches formant le corpus dans une *mémoire de masse* convenable, en général une pile de disques magnétiques, non sans avoir au préalable codé les informations de ces fiches en tops-tips et élaboré un programme pour remplir la mémoire de la façon la plus facile à exploiter.

Arrive un client qui voudrait se documenter sur tel sujet appartenant à celui des 200 grands domaines auquel le centre est consacré. Il rédige ses souhaits en clair sur une « demande » qu'il tend à un analyste. Qu'en fait celui-ci ? La demande est-elle prête à être confrontée au corpus ?

▶▶ — Certainement pas, puisque la demande formule en clair les thèmes sur lesquels le client cherche des références, alors que dans le corpus, le contenu des documents disponibles est caractérisé avec les seuls descripteurs du thesaurus.

▶ — Donc, je le répète, que doit faire l'analyste avant de s'adresser à l'ordinateur ?

▶▶ — Il transpose la demande du client en la caractérisant elle aussi avec les seuls descripteurs empruntés au thesaurus. Puis il inscrit ces descripteurs selon leur code top-tip sur, mettons, une carte perforée ou un carton.

▶ — Bravo. Il a maintenant une fiche de demande qui est écrite *dans le même langage* que les fiches du corpus, et sur un support que l'ordinateur sait lire. Et dès lors la vie est belle. L'ordinateur va confronter la fiche demande avec toutes les fiches du corpus qu'il fait défiler une à une et...

▶▶ — Il extrait celles dont l'étiquette descriptive coïncide avec la fiche demande.

▶ — Après quoi il fait imprimer par son imprimante les labels des fiches qu'il a ainsi extraites. Leur ensemble constituera bien la liste des indicatifs de tous les travaux existant dans la bibliothèque et traitant du sujet qui intéressait le client. Ce sera donc la liste des références bibliographiques que ce dernier cherchait. On s'est au fond ramené au *tri sélectif d'un fichier*, le fichier étant le corpus et le critère de sélection étant qu'il faut présenter les mêmes descripteurs que la demande. Et pourquoi fallait-il pour cela un ordinateur ? Parce que le tri aurait été infaisable à la main à cause, précisément, du phénomène qui a motivé ce chapitre : l'énormité de la littérature. Un corpus contient facilement des centaines de milliers de fiches, dont chacune est couverte de 5 à 20 descripteurs empruntés à un thesaurus qui en comprend 5 000 à 10 000. Pour les contraster une à une avec une fiche demande comprenant elle aussi une dizaine de descripteurs, quelconques parmi 5 000 à 10 000, il faudrait à la main un an de travail à une secrétaire... qui au bout d'une semaine serait devenue folle.

L'extraction documentaire par ordinateur peut être affectée de deux types d'erreurs qu'on appelle respectivement les « *silences* » et les « *bruits* ». Il y a « silence » quand une liste livrée par l'ordinateur ne contient pas des références qui auraient été

pertinentes, et il y a « bruit » quand elle contient des références qui ne le sont pas. Les silences sont en général dus à une insuffisante finesse dans l'indexation des documents. Par exemple, si dans un document sur la préhension des pattes des fauves, l'analyste a omis de mentionner qu'on y trouve des renseignements sur la rétractilité des griffes du tigre, un client qui vient demander de la bibliographie sur ce dernier thème ne recevra pas le document en question. En principe cette cause d'erreur peut être réduite si on veille à ce que l'indexation de la production littéraire soit complète. On peut aussi préparer des lexiques de connexion, que l'ordinateur consulte *automatiquement*, et où, sous les mots-clés « griffe », « tigre », « rétractilité », on trouve des *renvois* à « patte », « fauve » et « préhension ». Mais à mesure qu'on élargit le champ patrouillé par l'ordinateur pour satisfaire une demande, si les silences diminuent, les bruits augmentent. Car alors on risque de proposer au client des documents qui ne parlent de son sujet qu'accessoirement. Par exemple, le client ne s'intéresse, en fait, qu'à l'usage de l'acier dans les perceuses, et par le jeu de renvois automatiques vers des domaines plus généraux on lui sert un document sur la métallurgie des machines-outils, où il se trouve qu'on traite surtout du tungstène dans les fraiseuses. Ce document sera du bruit. Pour éviter ce second piège, le seul moyen est que le client, ou l'analyste qui le conseille, ajoute une série de précisions à sa demande, trace des frontières, prononce des exclusives. Par exemple, il demandera « acier pour outils mais pas métaux non ferreux », ou encore « acier pour outils si découpe ou dressage, non percement ». De plus, le client ne s'intéresse peut-être qu'aux travaux postérieurs à telle date, ou émanant de tels auteurs mais pas de tels autres, ou inférieurs à 20 pages, ou en français seulement. Toutes ces précisions qui cadrent mieux les références cherchées font l'objet de ce qu'on appelle une « *information ancillaire* », exprimée à côté de l'*information principale* par des inflexions logiques telles que « ou », « pas », « et », « mais », « seulement si », etc. Lorsque l'ordinateur a trouvé dans la mémoire de masse une fiche du corpus dont les descripteurs concordent avec ceux de la demande, il la recopie au vol dans sa mémoire statique, et pendant le temps qui s'écoule jusqu'au passage de la prochaine fiche du corpus, il effectue, à cadence ultra-rapide, les calculs permettant de voir si la fiche retenue dans le premier filtrage exécuté sur l'information principale, satisfait aussi à l'information ancillaire. Et ce n'est qu'au terme de ce second filtrage, s'il est positif, que le label du document est accepté comme référence bibliographique valable.

Quelques autres méthodes

Ce qui précède ne vous donne que l'ossature d'une recherche documentaire. Là-dessus se greffent bien entendu mille variantes et des raffinements parfois extraordinaires. Je ne vous en citerai que quelques-unes, pour vous montrer combien ce domaine est dynamique.

Certains laboratoires cherchent à réduire le rôle des analystes, voire à éliminer ceux-ci complètement, en mettant au point ce qu'on appelle l'« *indexation automatique* » des documents. L'ordinateur lit un document en entier (ou seulement son résumé) enregistre dans sa mémoire *tous* les mots, toutes les ponctuations, puis se met à compter combien de fois chacun intervient, combien de fois l'un se couple à d'autres,

ou suit un article, ou commence une phrase. Suivant des statistiques très savantes mises sur pied par le programmeur, l'ordinateur *décèle tout seul les mots-clés* et leurs liaisons principales. Il les compare alors au thesaurus, également enregistré en mémoire, et détermine les descripteurs qui décrivent au mieux le document. Inutile de dire que le programme qui instruit l'ordinateur d'exécuter cette sorte de découverte automatique des mots-clés qui conviennent à une œuvre, figure parmi les plus subtils et complexes qu'on ait entrepris de nos jours. D'ailleurs on n'en a encore dressé aucun de bon et l'indexation automatique, si on la veut de qualité, reste pour l'instant un rêve.

Dans une méthode moins (ou plus ?) ambitieuse, l'ordinateur se contente d'entrer en mémoire les seuls *résumés* des travaux, mais tels quels, en clair, avec tous leurs mots, en oubliant tout thesaurus. A raison de 100 mots par résumé, une bibliothèque déjà conséquente de, mettons 500 000 documents, n'aura jamais que 50 millions de mots à conserver en mémoire, ce qui tient sur une bande magnétique. On ajoute naturellement les indications signalétiques qui forment les *labels* des documents. Ceci étant fait, quand le centre documentaire reçoit une demande, un analyste y souligne les *mots-clés*, empruntés carrément au texte du client demandeur, et les présente (en code) à l'ordinateur. Celui-ci *relit* alors ***tous*** les résumés de tous les documents et ne retient que ceux qui contiennent *les mêmes mots* (ou des synonymes que l'ordinateur va chercher lui-même dans un dictionnaire enregistré dans sa mémoire). L'analyste ou le demandeur ajoutent encore des précisions ancillaires, mais en langage courant, comme par exemple : je veux seulement les documents dans les résumés desquels les mots « moustiques » et « blonds » figurent dans une même phrase mais ne soient pas adjacents, ou ceux dans lesquels les mots « métaux non ferreux » soient adjacents, ou ceux dans lesquels figurent les mots « achats en premières noces » mais pas les mots « immeubles » ni « titres », ou ceux dont les auteurs comprennent le mot « Einstein », etc. Cette « logique ancillaire » peut être facilement programmée sur la machine, de façon que la sélection des documents soit beaucoup plus fine et pertinente que si on comparait seulement les descripteurs. Elle permet, comme je vous l'ai dit, de préciser un cadre dans lequel on cherche les documents, mais cette fois en clair, sans avoir à passer par la lourde syntaxe symbolique d'un thesaurus.

Dans tous les modes d'organisation connus à ce jour des corpus et des thesaurus, les extractions documentaires restent « longues » en dépit de la prodigieuse vitesse de l'ordinateur. Une question de poids, dont la réponse implique, mettons, une liste de 200 références bibliographiques, exige, pour que cette liste sorte tout imprimée, dix à quarante minutes. Cette durée est surtout due à la nécessité de parcourir d'un bout à l'autre d'énormes mémoires de masse séquentielles. Il serait hautement irrationnel de faire cette excursion à neuf pour chaque demande. Aussi, dans les centres documentaires, groupe-t-on les demandes, mettons d'un ou deux jours, et on les traite toutes à la fois, une nuit, pendant une seule passe de la mémoire de masse. Ce type de service oblige naturellement le client à attendre un ou deux jours la réponse, mais ce n'est pas grave dans beaucoup de situations.

Un moyen d'aller plus vite, quand on a un corpus assez réduit et homogène pour qu'on puisse l'organiser un peu logiquement par thèmes, c'est d'utiliser un fichier « *inversé* ». Au lieu qu'*une* fiche soit consacrée à *un document* et porte dessus les descripteurs qu'on y rencontre, on consacre *une* fiche à *un descripteur* et on inscrit dessus tous les documents qui en parlent. On réalise alors très rapidement le premier filtrage en appelant tous les descripteurs d'une demande et sortant les numéros des

documents qui figurent sous tous à la fois. L'extraction devient si rapide qu'on peut la faire sur une console de visualisation qui se couvre en quelques secondes des références répondant à la logique « ET », c'est-à-dire à la coïncidence simple des descripteurs dans tous les documents cités. C'est *à la vue*, dès lors, que le client procède aux sélections secondes que prescrit l'information ancillaire. Le client converse avec l'ordinateur comme avec une vendeuse infiniment patiente qui lui propose des articles sans cesse renouvelés selon son goût ou sa critique.

Les méthodes tentées pour faire découvrir, par l'ordinateur lui-même, les mots-clés d'un texte, obligent à automatiser l'analyse sémantique et syntaxique, ce qui conduit très vite à la linguistique. Par là s'établit un pont avec une discipline voisine, la *traduction automatique*, où l'ambition est d'obtenir qu'un ordinateur engendre tout seul la version dans une langue d'un texte écrit dans une autre langue. Cette discipline eut ses heures de gloire dans les années 60, mais elle n'est plus étudiée aujourd'hui que sur un plan de recherche désintéressée. On n'a, en effet, jamais abouti à des traductions acceptables. La raison en est que le langage courant est de plusieurs ordres de grandeur, plus subtil, complexe, mouvant, polymorphe, que tout ce qu'on pourrait embrasser et légiférer dans un programme.

Le crime ne paie pas

Bien d'autres systèmes sont employés dans l'art très raffiné de l'extraction documentaire où se mélangent la mécanique, l'optique et l'informatique. Je ne vous en citerai plus qu'un pour terminer : le système qu'emploie la gendarmerie nationale pour rechercher les criminels.

Une fiche est réalisée sur *microfilm*. Dans la moitié inférieure on place une photo, ou un texte, ou les deux, décrivant en clair, mais en résumé, un individu « suspectable ». Dans la moitié supérieure on inscrit, au moyen d'un appareillage approprié, une série de points blancs disposés sur une série de lignes et qui, selon un code, jouent le rôle de « descripteurs ». Par exemple, une certaine constellation de points signifie : effraction par les toits; une autre : attaque de personnes âgées; une autre : s'intéresse surtout à l'argenterie; une autre : balafré; une autre encore définit une taille, une ville de prédilection, etc. Les caractéristiques des empreintes digitales, des semelles de souliers, de la prononciation, ont pu aussi être exprimées, grâce à de savants travaux de « classification des formes », par des rangées de chiffres eux-mêmes codables en constellations de points.

Tous ces « descripteurs » relatifs à l'individu fiché, à ses méthodes et à son palmarès, peuvent d'ailleurs s'accompagner d'un peu de syntaxe et d'indications ancillaires du type « n'opère pas de nuit *parce que* myope ». Ces descripteurs et ces conventions forment l'équivalent du « thesaurus » d'un centre documentaire, tandis que le fichier même forme l'équivalent du « corpus ». Ceci dit, supposons qu'un acte délictueux vient d'être repéré par la gendarmerie de tel lieu. Après une enquête diligente, celle-ci communique au Centre, par fil spécial, une description de toutes les caractéristiques du délit constaté et de tous les indices relevés, description exprimée selon les termes et les conventions qui forment le thesaurus en question. Au bout du fil, une machine spéciale, recevant ces descripteurs de l'acte commis, les imprime automatiquement sous la forme de petits points blancs sur la partie haute d'une microfiche.

Appelons « fenêtre » la constellation ainsi obtenue, équivalente à ce qui en recherche documentaire était la « fiche demande » d'un client (après qu'elle eût été transposée par l'analyste en langage de thesaurus). Cette fenêtre va maintenant être présentée par une machine à la partie haute de toutes les microfiches de tous les individus fichés au Centre. La machine fait défiler ceux-ci à très grande vitesse, et un système de rayons lumineux et de cellules photoélectriques, dont les signaux sont traités par ordinateur, sonde leurs parties hautes pour comparer les constellations qu'elles portent avec la fenêtre. Chaque fiche qui donne une coïncidence tombe automatiquement dans un panier, qui, à la fin de l'opération, c'est-à-dire au bout de 5 minutes, contiendra ainsi tous les individus pouvant « logiquement » être suspectés dans le délit rapporté, du fait que leurs caractéristiques coïncident avec celles du délit. On peut naturellement commander l'extraction des auteurs « possibles » du délit selon des critères plus fins ou plus larges à volonté, et restreindre aussi la fraction du corpus qu'on explore. En dehors de la recherche de suspects fichés, compatibles avec les indices recueillis, le corpus et le thesaurus permettent d'importantes études de criminologie, en particulier pour examiner s'il existe ou non des corrélations entre l'âge, la nature des délits, l'éducation, la récidive, le statut de la famille, etc. Laquelle science en retour permet aussi de mieux circonscrire les recherches.

En conclusion, lorsque vous préparerez un hold-up, prenez soin de changer toutes vos habitudes et marchez en chaussettes.

Philosophie de la documentation automatisée

Vous voyez que l'ordinateur a fait une entrée fulgurante dans le domaine de la documentation et que ce domaine à son tour est en fait beaucoup plus vaste que la simple recherche de références bibliographiques. Nous venons de voir qu'il embrasse des activités comme la recherche des criminels. Il en embrasse encore bien d'autres, dont je ne vous cite que quelqus-uns : — la recherche de la jurisprudence, où il s'agit de retrouver toutes les sentences prononcées dans des procès et qui présentent un certain nombre de descripteurs en commun avec le cas traité, et où l'ordinateur promet de fournir enfin des listes vraiment exhaustives et pertinentes (mais où la difficulté majeure est de faire entrer dans le corpus les 120 tomes gros format qui enferment la jurisprudence passée); — la recherche des hommes ou des entreprises sachant faire telle ou telle chose, ou des sujets d'étude qui sont en activité dans tous les laboratoires d'un pays, ce qui suppose qu'on dispose d'un corpus complet sur les experts disponibles, les spécialités étudiées, les travaux en cours; — la recherche des valeurs numériques qu'ont toutes sortes de paramètres (le taux de natalité d'un pays, la température de fusion du fer, la production charbonnière d'un bassin, la fraction du budget couverte par la TVA, les surfaces des terrains au cadastre, etc.) dans des corpus contenant ces données enregistrées sous divers descripteurs, corpus qu'on appelle de plus en plus des « banques de données »; — la recherche des propriétaires des plaques minéralogiques, ou des bigames, ou des familles de plus de 8 enfants, etc.

Pour terminer ce chapitre, réfléchissons un peu sur ce qui a rendu possible la recherche documentaire par ordinateur. C'est au fond que dans les productions

humaines — celles du moins qui visent des objectifs et emploient des moyens « raisonnés » — on peut distinguer ***ce de quoi on parle*** et ***ce qu'on en dit***. L'ensemble des « descripteurs » caractérise en effet *ce de quoi on parle* et qu'on pourrait appeler le « sujet » traité en prenant ce mot dans le même sens qu'en grammaire, « sujet d'une phrase ». Et *ce qu'on en dit*, c'est-à-dire le texte même du travail que la documentation automatique va sortir, joue le rôle de ce qu'en grammaire on appelle le « prédicat » énoncé à propos du sujet. Il s'agit là en vérité d'une des plus grandes et plus fondamentales catégories du rapport entre l'homme et l'univers. Car le « sujet » à son tour, ou « ce de quoi on parle », est une forme réduite du monde extérieur, tandis que le « prédicat » ou « ce qu'on en dit » joue le rôle du « moi pensant » en face de ce monde, de ses réactions et recherches. Et ces deux catégories se distinguent de façon frappante surtout en ceci que le monde extérieur, donc le « sujet » du discours éternel et global de l'humanité, se laisse décrire par une série de caractéristiques objectivables, tandis que le monde intérieur des commentaires a des inflexions infiniment changeantes avec l'être qui les conçoit. Un problème de recherche documentaire consiste à retrouver tous les commentaires variables qui ont été faits sur un sujet fixe. Les sujets sont codables par des symboles, de même que le monde extérieur se laisse décrire par des paramètres objectifs; tandis qu'avec le commentaire, les seules choses codables qu'on puisse trouver sont les reflets qu'y projette le sujet. Vous avez reconnu d'une part les descripteurs qui caractérisent le document et d'autre part les descripteurs qui caractérisent la demande. Cette dichotomie est la clé de toute l'affaire. Une fois de plus, ce que nous retrouvons au fond des machines apparemment les plus supra-humaines montées par l'homme, c'est lui-même.

On a d'abord pris la mesure du problème que pose l'accès à des renseignements ou travaux qu'on recherche, dans une civilisation ployée sous des montagnes de connaissances. L'on a vu que, pour pouvoir frayer des chemins vers les documents cherchés, chacun doit être caractérisé d'abord par une *étiquette signalétique,* ensuite par une *étiquette analytique* faite de *descripteurs,* c'est-à-dire de mots-clés qui caractérisent son contenu et qui sont empruntés à une liste appelée « *thesaurus* ». Ces étiquettes codées, dont l'ensemble forme le « *corpus* », sont mises dans une mémoire et quand un client vient chercher les documents qui traitent d'un thème donné, un analyste écrit les descripteurs qui caractérisent ce thème et les présente à l'ordinateur qui va chercher dans sa mémoire, et en sort, les étiquettes signalétiques de tous les documents dont l'étiquette analytique contient les mêmes descripteurs. Après avoir exposé ce principe, on a vu son application à des problèmes variés tels que la recherche des criminels et la documentation automatisée.

11

Les applications « en temps réel »

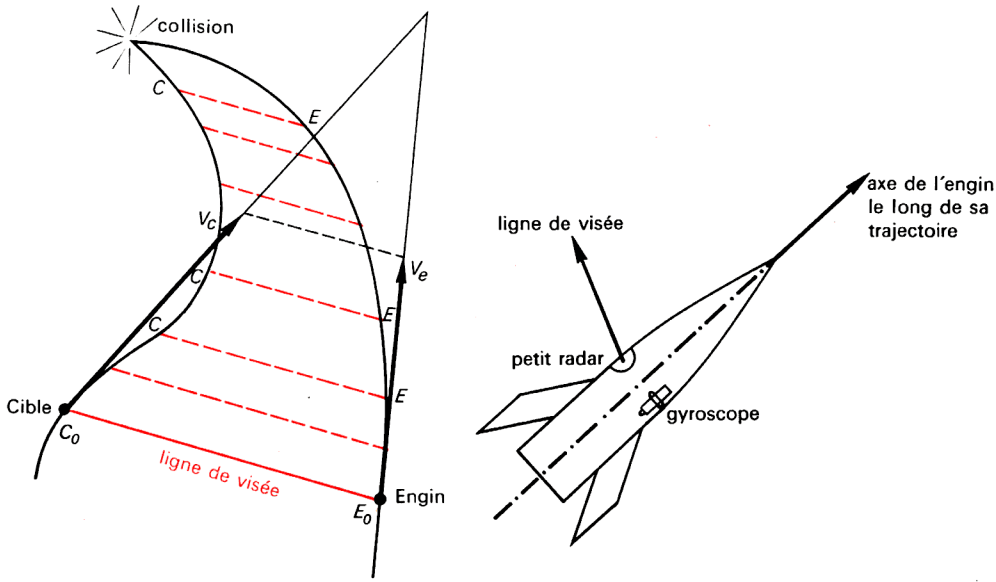
Le terme américain « **on line** », qui veut dire littéralement « en ligne », s'emploie pour désigner des équipements ou opérations qui s'insèrent en série dans un processus complexe et qui doivent donc recevoir leurs données et exécuter leurs programmes en même temps que les événements se déroulent, et avoir terminé chaque phase de leur tâche au moment où l'ensemble des processus le demande. Par contraste, on appelle « **off line** » les équipements qui travaillent en dehors d'un flux d'événements et résolvent en coup par coup des problèmes séparés, dans les temps qu'ils veulent. En français, on appelle ces deux modes d'intervention « *en temps réel* » et « *en temps différé* ». Jusqu'à présent, vous n'avez connu que la situation « en temps différé ». L'ordinateur trônait dans son hall, on venait lui apporter un problème tout huilé et ficelé comme une offrande à Baal, il l'avalait, le traitait, vous crachait sa réponse, puis se taisait, attendant le prochain problème qui pouvait être complètement différent. Toute autre est la situation dans quelques cas que nous allons examiner, empruntés tant à des tâches militaires qu'à des tâches industrielles.

L'interception

Comment un engin porteur d'une charge d'explosif fait-il pour aller frapper au but une cible qui se déplace presque aussi vite que lui, et en dépit des esquives qu'elle peut tenter? C'est le problème essentiel de l'interception en vol d'un mobile hostile, avion ou fusée.

Supposons qu'à un certain moment, l'engin ait pu parvenir en un point E_o tel que les deux vecteurs vitesse V_E et V_C de lui et de la cible forment un triangle plan avec la ligne de visée E_oC_o qui le joint à la cible. (Attention, la ligne de visée n'est *pas* située le long des trajectoires, ni celle de E ni celle de C, mais en travers d'elles.) Après ce moment la cible continue à voler, l'engin continue à la poursuivre, et la ligne de visée, c'est-à-dire la droite qui les joint, se déplace constamment dans l'espace. Mais si l'engin peut diriger son vol de façon que cette ligne, tout en bougeant, reste perpétuellement *parallèle à elle-même*, on démontre que l'engin rencontrera nécessairement la cible, quelles que soient les évolutions de celle-ci, à la condition toutefois que la vitesse de l'engin soit supérieure à celle de la cible.

Il s'agit donc d'asservir le vol de l'engin de façon que la droite EC qui le joint à la cible, et qui se déplace continuellement, reste néanmoins parallèle à sa position initiale E_oC_o. Il faut bien sûr pour cela que l'engin voie sans cesse cette cible, et



connaisse sans cesse sa propre direction. Pour voir la cible, il utilise un petit radar dit « de poursuite » qui, une fois qu'il a aperçu la cible sur son axe, maintient cet axe automatiquement pointé sur elle de sorte que cet axe matérialise la ligne de visée. Pour savoir à bord de l'engin vers où il vole, c'est-à-dire vers où pointe son propre axe (par rapport auquel celui du radar est en travers) on utilise un instrument appelé « gyroscope ». C'est ici que se place la difficulté qui nous oblige à recourir à l'ordinateur. Le gyroscope informe l'engin sur la direction de son axe dans l'espace, telle que la verrait un observateur immobile au sol, ce qu'on peut appeler la direction « absolue ». Tandis que le petit radar de bord, étant porté par l'engin, fournit seulement l'angle entre la ligne de visée et le corps de l'engin. Comme celui-ci virevolte dans l'air, la direction définie par cet angle n'est pas « absolue », ce n'est pas celle qu'un observateur immobile assignerait à la droite EC. Or c'est la direction absolue de EC qu'il faut maintenir parallèle à elle-même. On doit donc la *restituer* en fonction du cap absolu de l'engin et de l'angle que fait la ligne de visée par rapport à l'engin, données qu'on possède. Mais les formules à appliquer pour cela sont complexes et le calcul, s'il était fait à la main, durerait une heure. C'est ce calcul de « changement de système de référence » qu'on demande à un ordinateur miniature installé à bord d'exécuter en un centième de seconde. Aussitôt déterminée la direction absolue de la ligne de visée EC, il la compare à celle, E_oC_o, qu'elle avait au début, et commande des gouvernes qui infléchissent la trajectoire de E de façon à *diminuer* l'écart pris par la direction de EC. Le cycle se répète jusqu'à ce que l'écart soit annulé, c'est-à-dire que EC ait été ramené à être parallèle à E_oC_o. Tout cet asservissement du vol de l'engin par la condition que la droite EC reste parallèle à elle-même s'accomplit ainsi automatiquement, et si vite qu'on peut dire que pratiquement la condition reste satisfaite en permanence, d'où la certitude qu'il y aura collision quelque esquive que tente la cible.

Quant à amener l'engin depuis le sol jusque dans une position initiale favorable où « le triangle des vitesses se ferme », cette tâche est assurée par un autre ordinateur, placé cette fois à poste fixe sur le sol. Cet ordinateur aussi travaille « en temps réel ».

Gérer le radar

Il n'y a plus aujourd'hui de « système d'armes » qui ne soit farci d'ordinateurs travaillant « en temps réel ». Et ces ordinateurs sont souvent énormes, comme dans l'exploitation du radar. Vous savez certainement qu'un radar tournant permet de faire paraître sur l'écran d'un tube cathodique un petit spot lumineux en un point qui est homologue à l'endroit du ciel où se trouve un corps solide réfléchissant tel qu'un avion (*).

Pendant longtemps on se contentait de placer devant ces écrans des soldats spécialement entraînés qu'on appelait des « observateurs électromagnétiques ». Ils pointaient au crayon les spots qui apparaissaient, sur un transparent couvert de lignes coordonnées, et annonçaient les coordonnées à la voix à d'autres soldats (les Anglais utilisaient dans cet emploi des « soldates » fort avenantes). Ces assistants, ou assistantes, placés autour d'une immense table, déplaçaient à l'aide de bâtons de croupier des petites pièces de bois diversement colorées et sculptées, représentant les avions, selon les indications des « pointeurs électromagnétiques ». Cela formait la « carte du ciel » que les officiers, du haut de leurs loges, contemplaient d'un air méditatif. De temps en temps, ils téléphonaient des ordres à des escadrilles ou à des avions déjà en l'air.

Ce ballet des observateurs à terre devint impossible quand le ballet des aviateurs dans l'air se fit plus dense et plus rapide. Un ciel de bataille moderne contient facilement 200 avions se déplaçant à 1 000 km/h. Dès lors, l'assemblage du tableau de situation à la main et à la voix devenait beaucoup trop lent, et il fallut recourir à l'ordinateur. Pour faire quoi, exactement ?

Dans un premier temps, on a conservé l'observateur électromagnétique humain pour la raison suivante : l'écran d'un radar n'est pas seulement étoilé par les spots des avions, il contient encore de nombreuses autres taches ou traînées lumineuses dues au « bruit de fond » et aux échos parasites renvoyés par des nuages, par des oiseaux ou par le sol. L'homme est bien outillé, par son organisation mentale et visuelle, pour « reconnaître » les spots significatifs dans cette « soupe de chien » comme disent les techniciens. On le laisse donc faire cette besogne, mais on le munit d'un photostyl, ou d'un manche à balai, qui engendrent automatiquement les coordonnées de chaque point qu'il désigne, sous la forme de signaux électriques codés. Ces coordonnées vont entrer dans la mémoire d'un ordinateur qui possède ainsi, dans une grande « table », tous les plots désignés par les pointeurs (on appelle « plot », en radar, un spot muni de coordonnées et déclaré utile).

A partir de là il faut programmer l'ordinateur pour accomplir une série impressionnante de travaux pendant le temps même où les événements se déroulent. Le but général est de retransmettre ces plots sur l'écran d'une autre console qui va remplacer l'immense table de jadis et devant laquelle se tiendra l'officier avec, autour de lui, une panoplie de boutons, de téléphones et de panneaux qui ferait pâlir d'envie les metteurs en scène des grands pontes de l'an 2000. Mais ces plots seront « habillés », c'est-à-dire que l'ordinateur, avec des renseignements intégrés à partir de plusieurs

(*) Vous apprendrez comment ce résultat est possible dans le volume « Comprendre l'électronique » de la présente collection.

postes d'observation et selon des algorithmes d'estimation très subtils, décide, et affiche sur l'écran par des symboles idoines, si le plot est ami ou ennemi, s'il représente un avion isolé ou un raid d'avions groupés et dans ce cas de quelle force, à quelle altitude, à quelle vitesse. La détermination de la vitesse n'est d'ailleurs qu'une partie d'un travail plus conséquent qui est d'identifier les trajectoires, ou « pistes », à travers la constellation embrouillée des plots qui s'éteignent et se rallument ailleurs, à chaque tour d'antenne; tâche à laquelle l'organisation mentale et visuelle de l'homme se prête mal.

L'ordinateur s'y entend mieux. Tous les plots recueillis au tour d'antenne N° 1 sont, rappelez-vous, déposés dans la mémoire, dans cette partie de la mémoire qu'on appelle la « table ». Sur l'écran leurs images vont s'évanouir graduellement à mesure que s'écoulent les 8 secondes d'un tour et être remplacées, au tour d'antenne N° 2, par une nouvelle emblavure de plots frais légèrement décalés des précédents, puisque les avions auront bougé. Dans la table, cela se sera traduit par l'arrivée d'une nouvelle génération de plots. Là il n'y a personne pour voir un passage graduel de chaque plot à un plot voisin. On a simplement 200 plots P_1 recueillis au tour d'antenne N° 1, c'est-à-dire 200 coordonnées, et 200 plots P_2, donc 200 autres coordonnées, recueillis au tour d'antenne N° 2. Le seul moyen pour déceler des pistes est d'associer chaque P_1 avec chaque P_2 — ce qui fait $200 \times 200 = 40\,000$ associations à essayer — et de voir si les deux plots d'une paire peuvent plausiblement appartenir à un même avion qui serait passé d'une position à l'autre pendant les 8 secondes d'un tour d'antenne. A cet effet le programme commande à l'ordinateur de calculer la distance entre les deux plots appairés et de décider qu'ils ne peuvent pas appartenir au même avion, donc à une même piste, si cette distance est supérieure à celle qu'aurait pu franchir en 8 secondes l'avion le plus rapide existant à l'époque. Les paires de plots qui ne sont pas exclues par ce test sont considérées comme autant d'amorces possibles de pistes. Pour vérifier si elles correspondent vraiment à des pistes, l'ordinateur calcule la zone de l'espace où devrait se retrouver un avion, au 3e tour d'antenne, c'est-à-dire encore 8 secondes plus tard, s'il avait volé du premier au second plot de la paire et continué en ligne droite. Vu les erreurs inévitables qui affectent les mesures, cette zone extrapolée possède une petite étendue et s'appelle de ce fait une « fenêtre d'attente ». Toutes les fenêtres d'attente pour le 3e tour, pour toutes les paires retenues entre les tours N^{os} 1 et 2, doivent être calculées entre les tours N^{os} 2 et 3 afin d'être prêtes au tour N° 3. Si, au tour N° 3, aucun plot ne paraît dans la fenêtre extrapolée à partir d'une paire P_1P_2 retenue d'après les tours N^{os} 1 et 2, on décide que cette paire ne correspondait pas à une vraie piste, et on l'élimine. Si au contraire un plot y paraît, on adopte comme confirmée la piste constituée par les trois plots qui concordent. L'ordinateur fait afficher un symbole adéquat pour en informer l'officier, puis se met à « entretenir », comme on dit, la piste qu'il a détectée, en calculant constamment par extrapolation la fenêtre d'attente dans laquelle le plot de l'avion poursuivi doit apparaître au tour suivant. Si plusieurs fois de suite des fenêtres d'attente ainsi calculées ne sont pas remplies, l'ordinateur décide que la piste qu'il avait suivie jusque-là était fausse, ou a disparu, et l'efface de ses tables. Mais il n'oublie pas de patrouiller perpétuellement les plots nouveaux qui pénètrent dans le ciel, pour les appairer et amorcer de nouvelles pistes.

Voilà déjà, n'est-il pas vrai, des secondes bien remplies par un programme d'essais et des calculs considérables. Et ce n'est encore qu'une petite partie du travail

à faire. Il faut en outre prévoir des consignes de calcul pour de nombreux cas pathologiques, tels que trajectoires qui s'incurvent ou se croisent, plots représentant un raid groupé qui soudain éclate en plusieurs pistes, etc. Il faut en outre donner des numéros aux pistes, afin que l'ensemble des échelons de commandement et d'action qui aura à intervenir, puisse se référer vite par un même nom à une même piste. Ces numéros doivent se renouveler et valser correctement à mesure que les pistes meurent et naissent. Il faut identifier, par une série de tests, les amis et les ennemis, les bombardiers et les chasseurs, et instruire les consoles de présenter auprès des plots les symboles choisis pour signifier ces deux faits. Il faut pouvoir fournir des coupes du ciel sur demande, par exemple présenter à un officier tous les avions qui se dirigent vers un certain secteur, ou tous ceux qui dépassent une certaine altitude. Il faut en outre tenir à jour l'état de toutes les bases, leur armement disponible, la direction de leur piste d'envol, leur moral, et calculer en permanence comment se présenterait et se déroulerait une tentative d'interception contre tels objectifs si on la lançait de telle base à l'instant qui passe, cela afin d'aider le commandement à prendre la meilleure décision. Et ce « calcul d'assignation d'armes » suppose qu'un vaste flot d'informations codées circule entre toutes les bases aériennes et tous les radars. Il faut faire savoir à tous, par l'apparition de symboles adéquats sur les écrans radars, quand un objectif a été engagé, afin de ne pas doubler les attaques. Puis il faut guider l'avion intercepteur vers le meilleur point pour le lâcher des engins, en effectuant la grosse masse de calculs que cela comporte. Après un combat, il faut prendre en charge l'intercepteur ami, s'il a survécu, et le diriger vers la meilleure base d'accueil, qu'il faut calculer en fonction de l'endroit où le combat s'est livré, tout en signalant à toutes les consoles que tel spot est un ami qui rentre. Il faut encore tenir compte de la météo, qu'il faut afficher partout, des fréquences disponibles pour les communications radio avec les pilotes en l'air, etc. Et tout cela, rappelez-vous, « *on line* », pendant que les événements se déroulent.

Ces tâches sont en vérité si gigantesques que même avec les plus puissants calculateurs modernes, il en faut plusieurs pour les accomplir, en général trois : un calculateur pour dépouiller le ciel, un pour aider l'assignation des armes, un pour guider les interceptions et les recueils. Et on ne s'est pas encore limités à ces ambitions. Avec des avions de plus en plus rapides et nombreux, la désignation initiale des spots utiles par un homme dans la « soupe de chien » devient à son tour trop lente et trop incertaine. On dresse alors un super-programme pour faire ce qu'on appelle « l'amorçage automatique » des pistes. Ce programme travaille sur les signaux électriques eux-mêmes produits par le radar en arrière des écrans cathodiques. On digitalise ceux-ci, et on les soumet à une série savante de tests très subtils pour estimer s'ils émanent d'avions ou de bruits ou d'échos parasites. Ce travail exige une masse de calculs encore plus considérables que tous les précédents, et dont la programmation n'est toujours pas complètement maîtrisée aujourd'hui. Dans les installations qui en sont dotés, les pointeurs humains disparaissent tout à fait, et les écrans se peuplent tout seuls des plots et pistes issus de l'ordinateur chargé des amorçages.

Le radar a été l'une des forces motrices essentielles du développement de l'ordinateur. C'est lui qui a exigé des machines de plus en plus puissantes et performantes. C'est de lui qu'est sortie la console de visualisation sur écran cathodique. C'est lui encore qui a éperonné les programmeurs. J'ai été personnellement engagé dans ces efforts, et j'ai constaté qu'après quelques mois de travail avec mon équipe, le plus

difficile était de se rappeler ce qu'on avait conçu, ce qu'on avait demandé à la machine de faire : à quel poste avions-nous décidé que serait envoyée quelle information à quel moment ? Qu'en faisait-il et quel nombre de microsecondes avions-nous estimé que cela consommerait ? Nous dûmes nous confectionner un catalogue des actes et liaisons que nous avions envisagés, et, six mois après, il fallut dresser un catalogue pour pouvoir consulter ce catalogue... On sent ici que l'une des limitations de l'ordinateur est la capacité mentale du ou des programmeurs...

Vous pensez bien qu'une fois que les forces aériennes eurent maîtrisé cette technique pour présenter automatiquement le ciel de combat et prévoir les effets des interventions, les marins et les fantassins n'allaient pas rester en parents pauvres. On développe aujourd'hui des systèmes qui assurent aussi la surveillance d'un champ de bataille terrestre et l'estimation calculée des effets des manœuvres qu'on peut y déployer. On rassemble des informations à partir de toutes les sources possibles : des estafettes qui téléphonent, des postes munis de longues-vues et de télémètres, des détecteurs de bruit, de fumées, de rayonnements, de vibrations, enterrés ou suspendus en de multiples endroits, des ballons qui télévisent le sol, des radars sensibles aux autres radars, des rideaux de rayons infrarouges, etc.

Tous ces appareils d'observation transforment automatiquement et instantanément ce qu'ils perçoivent en signaux électriques codés, lesquels convergent par fils ou ondes radio vers un gros ordinateur placé dans un très gros camion. Et cet ordinateur fait sur ces données les genres de calculs que nous venons de décrire à propos des combats aériens, encore plus compliqués parce que les situations et actions possibles sur terre ferme sont de un ou deux ordres de grandeur plus variées et complexes que celles qu'on peut rencontrer ou déployer dans le ciel.

Ainsi se renverse le vieil adage de Napoléon, qui disait : « On s'engage et on voit ». Désormais « on voit et on s'engage. » Et c'est infiniment plus humain. Car s'engager d'abord, ça consistait en fait à sacrifier des hommes uniquement pour pouvoir voir, comme les primitifs sacrifiaient des vies pour sonder l'humeur des dieux. Aujourd'hui on peut, grâce à des réseaux d'instruments dont les mesures sont récoltées, codées et intégrées par l'ordinateur, reconnaître le terrain en détail et supputer les effets des actions avant de faire couler le sang. Il était d'ailleurs urgent qu'on pût faire « en temps réel » ces calculs épargnant des vies, car les armes modernes en fauchent bien davantage que jadis à chaque engagement. Dans l'enfer moderne de la détestable guerre, l'ordinateur donne ainsi la possibilité d'apporter, il importe de le noter, une petite touche d'humanisation.

L'ordinateur dans l'industrie

L'ordinateur intervient de plus en plus « en temps réel » dans les processus industriels. Déjà au niveau de l'atelier de mécanique traditionnel, on peut l'utiliser pour bien des tâches que jusqu'ici on n'exécutait que par des moyens lourds et chers. Un exemple est le problème de tailler une pièce de métal selon un profil donné. Quand ce profil était trop tarabiscoté pour être copié à la main, on utilisait des machines complexes de duplication dans lesquelles un palpeur suivait le profil préalablement réalisé à la main, sur bois ou en plâtre, et entraînait par des jeux complexes d'engre-

nages les commandes qui font s'enfoncer plus ou moins dans la pièce métallique le couteau d'un tour ou d'une fraiseuse. Non seulement il fallait réaliser le modèle en bois à force d'infinies retouches, mais encore la duplication n'était jamais assez fidèle à cause de l'inévitable jeu dans les engrenages, et il fallait finir le surfaçage à la main. Au lieu de ça on peut donner à un ordinateur l'équation mathématique du profil à réaliser, et en chaque position qu'atteint l'outil, en un centième de seconde, il calcule la distance que celui-ci doit franchir pour aller s'appliquer exactement à l'endroit voulu. Après quoi, il envoie des signaux qui commandent l'engagement de l'outil en conséquence. L'ouvrier n'est plus là que pour surveiller. On vend aujourd'hui des machines-outils avec des ordinateurs incorporés.

Ce n'est là qu'une application très simple. On peut faire bien plus.

La commande des machines-outils par ordinateurs, de plus en plus répandue, n'est qu'un cas particulier de ce qui précède. Au lieu qu'un ouvrier surveille sa pièce d'un regard tendu et enfonce ou tourne son outil plus ou moins selon le profil souhaité, l'équation mathématique de celui-ci est communiquée à un ordinateur, qui reçoit d'autre part continûment, pendant le travail de la machine, des informations sur la position de l'outil par rapport à la pièce. Comparant cette position au profil à réaliser, l'ordinateur en déduit, grâce à un programme, les ordres à donner pour amener l'outil où il faut. On parvient ainsi à automatiser entièrement des usinages complexes.

Si on combine toutes ces méthodes avec des critères régissant au mieux l'ensemble des activités d'une usine (à quel moment faut-il faire venir des pièces du magasin, où faut-il les entreposer, quelles mesures faut-il prendre, quels modèles permettront d'exploiter au maximum la matière première à mesure qu'elle se débite, etc.), on arrive à des programmes de *production optimum*. Quelques pas de plus dans le même sens, et on pourrait déboucher sur des éléments de planification économique. Mais ces derniers pas sont si difficiles qu'on ne les verra sans doute abordés que dans bien des décennies.

Considérons une activité qu'on pensait jusqu'ici réservée à des dispositifs de petite mécanique directement animés par la main et l'esprit de l'homme : la composition typographique. Il faut arranger les « caractères » dans les mots, les mots dans les phrases, les phrases dans les pages, de façon que le tout, une fois imprimé, soit élégant, aéré, muni de sous-titres, figures, indices, alinéas appropriés, et respectant la sacro-sainte « justification », c'est-à-dire que toutes les lignes se terminent exactement sur la même verticale à droite, de la même façon qu'elles commencent à gauche. Bien que ces opérations ne paraissent exécutables que grâce à l'intelligence et au doigté d'un artisan habile, comme la confection d'un dessin, on a réussi à écrire des programmes qui permettent à un ordinateur de les régler.

Il faut d'abord, pour cela, rendre le maniement physique des signes plus léger, plus souple que l'ancien maniement de caractères en « plomb » (le terme a subsisté bien qu'il s'agisse d'un alliage de plomb, d'antimoine et d'étain). Vous savez sans doute que la méthode d'impression dite « *offset* » utilise non plus le plomb, qui avait régné en maître dans l'imprimerie depuis Gutenberg, mais des plaques appelées « reports » (en anglais, *report* se dit *offset*), où les parties imprimantes sont constituées par un tracé *plat* qui « accroche » l'encre, laquelle ne marque pas sur les parties environnantes. Ces parties imprimantes sont reportées sur la plaque par des procédés physico-chimiques simples à partir de textes (dans l'exemple considéré) portés par des films transparents appelés « *typons* ». Tout revient donc à préparer les typons.

A cet effet, on peut évidemment partir du texte composé, à la main ou mécaniquement, avec des caractères métalliques. Mais, pour opérer avec la souplesse que requiert un contrôle *ordino*, on prend comme support initial de l'écriture la face d'un tube cathodique, comme dans les consoles de visualisation dont je vous ai parlé. Sur cet écran, le faisceau cathodique dessine les caractères à sa super-extravitesse, comme nous l'avons déjà dit. La suite des points qui forme le tracé des différents caractères est enregistrée en mémoire, et on aura autant de suites qu'il y a de *types* de caractères entre lesquels on veut pouvoir jouer (minuscules, majuscules, gras, romain, italiques, elzévir, gothique, etc.). Cette « *mémoire des caractères* » est en général constituée par une pile de disques ou un tambour magnétique, afin qu'on puisse accéder à chaque caractère en une infime fraction de seconde. C'est une unité fixe de l'installation, montée et remplie une fois pour toutes. Une caméra, placée devant l'écran, photographie les caractères que dessine le faisceau cathodique et les transpose ainsi sur un film qui va constituer le typon. Tel est, en très grandes lignes, le *support physique* de la nouvelle typographie en photocomposition. Vous voyez que la facilité avec laquelle on peut le modifier au vol permet de le soumettre à un contrôle *ordino*.

Comment marche ce contrôle? Une claviste tape le texte sur une machine à écrire électrique qui communique les codes successifs de tous les signes touchés à une mémoire statique (ou une bande magnétique). Cette seconde mémoire s'appelle la « *mémoire du texte* », et son contenu varie avec chaque texte. La claviste n'exécute elle-même aucune composition. Elle tape, comme on dit, « au kilomètre », mais en indiquant au passage, au moyen d'un panneau de « commandes typographiques », les types de caractères et la mise en page souhaitée (longueurs des lignes, interlignes, marges en haut et en bas, etc.). Lorsque tout le texte est ainsi entré en mémoire, c'est un programme interne qui prend le relais; et, à partir de là, nous vivons au rythme *ordino*.

Le programme commence par faire compter à l'ordinateur les longueurs des lignes qui se formeraient et qu'il fait comparer aux instructions de mise en page. Si une ligne est telle que son dernier mot complet dépasse la longueur prescrite, le programme essaie, soit de renvoyer le mot « en trop » à la ligne suivante et de reconstituer une ligne justifiée *sans* ce mot en modulant les intervalles entre caractères dans les limites prescrites, soit, si cette manœuvre échoue, de couper ce mot « abusif ». Pour ce faire, le programme doit consulter une troisième mémoire, qui n'est autre qu'un dictionnaire de la langue française indiquant les coupures syllabiques correctes. Après quoi le programme considère la ligne comme définitivement arrêtée, et la transfère dans une quatrième mémoire, dite *de transit*. Le restant du mot coupé devra être comptabilisé dans la ligne suivante. Quand toutes les lignes d'une page, ou d'un paragraphe, ont été ainsi préparées *ne varietur* avec tous les codes de tous les caractères logés dans la mémoire de transit, le programme donne à celle-ci l'ordre de faire sortir ces derniers de la mémoire des caractères vers l'écran, où ils se dessinent avec le type et la mise en page correct, et d'où ils sont repris par la photo pour engendrer le typon.

Vous voyez que le programme de composition est une grosse affaire, bourrée de branchements conditionnels, d'essais et contre-essais. Il grossit encore lorsque, comme c'est souvent le cas, on veut en plus pouvoir traiter des *corrections d'auteur* éventuelles, sans perdre la composition des parties du texte qui ne sont pas affectées par la correc-

tion. Car il faut dans ce cas, après avoir sorti les typons pour permettre aux auteurs de se relire, insérer dans le texte ancien des parties dont la longueur n'est jamais égale aux parties modifiées, ce qui exige de nouveaux calculs savants de remise en page. Cependant, tout énorme qu'il soit, un programme de composition ne s'écrit qu'une fois; c'est un investissement de base qu'on fait en même temps qu'on achète les machines. Et une fois cet investissement fait, on peut réduire la durée de la composition de près de 5 fois par rapport au travail manuel, la partie la plus longue de la méthode *ordino* étant d'ailleurs celle qui doit encore se faire manuellement, à savoir la première frappe du texte (frappe « au kilomètre »). Une installation puissante servie par un programme efficace peut composer 32 pages de journal en moins d'une demi-heure après la frappe du texte. L'ordinateur est ainsi en train de révolutionner l'une des industries les plus vénérables et qui paraissait intangible : l'imprimerie.

On va encore plus loin dans les applications « en temps réel ». Une usine qui distille le pétrole et qui en tire, en plus de l'essence, les mille dérivés qu'on emploie en chimie, pharmacie ou parfumerie, n'est après tout qu'un vaste *algorithme animé*, qui opère sur des produits incidents jouant le rôle de *données* pour engendrer, après une série de manipulations réglées, d'autres produits ou articles jouant le rôle de *résultats* ou solutions. Si aucune des manipulations ne peut jamais donner lieu à aucune incertitude quant au pas suivant, on peut en principe faire exécuter leur série complète par des mécanismes astucieux, agrémentés de chimie, qui mènent leur jeu tout seuls selon un programme, chacun intervenant à point nommé pour faire faire au processus un pas élémentaire tel que : transporter, réduire, filtrer, coller, mélanger, dissoudre, évaporer, condenser, peindre, etc. Dans les processus complexes on peut avoir besoin de rouages très fins, subtils, précis ou rapides, empruntant à toute la gamme des moyens d'action de la technique humaine. Mais, à la difficulté près de mettre au point ces rouages, on devrait toujours pouvoir arriver au bout, avec très peu d'interventions humaines, ou pas du tout. On aboutit ainsi aux fameuses chaînes « automatiques » de l'industrie moderne, qui ont émerveillé les profanes bien avant l'ordinateur.

Mais en général on ne peut pas se fier à un algorithme de production défini d'avance d'un seul jet, parce que les impuretés et imprécisions inévitables des produits et des instruments peuvent provoquer en cours de route des écarts importants par rapport à la marche idéale; des écarts tels que, si on n'en tient pas compte, tout le processus déraille. Souvent on peut faire rentrer les choses dans l'ordre en plaçant aux bons endroits des *correcteurs automatiques*, qui mesurent l'écart et en déduisent en retour une action renforçante ou affaiblissante sur telle ou telle phase, action qui s'exerce toute seule jusqu'à ce que ledit écart s'annule. Ces systèmes, qui ne sont en équilibre que tant qu'une certaine grandeur utile reste égale à une valeur idéale et qui s'asservissent automatiquement sur cette valeur, s'appellent des « *servo-contrôles* ». D'un processus parsemé non seulement de mécanismes automatiques mais encore de servo-contrôles, on dit qu'il a été traité par « *automation* ». Vous savez que c'est là un maître mot de la technique moderne.

La phase suivante dans la sophistication croissante est celle où les écarts qui se produisent dans le processus industriel, ou ce processus même, appellent non plus seulement des auto-corrections, mais *plusieurs chemins d'action différents*. C'est l'occurrence que, dès le Chapitre 1, nous avons appelée un « **branchement** ». Dans une raffinerie de pétrole on doit faire d'innombrables tests sur les fluides qui circulent à

divers endroits et à diverses époques, et, selon les résultats de ces tests, le fluide doit être mélangé à d'autres produits, ou dévié en tout ou en partie vers d'autres tubulures, ou chauffé, ou refroidi. Le programme comprend donc nécessairement ce que nous avons appelé des « *ordres et branchements conditionnels* ». Or les servo-contrôles *continus* de l'automation sont incapables de commander ces branchements conditionnels. C'est alors qu'intervient l'ordinateur. A condition qu'on lui communique toutes les données caractérisant l'état des fluides à tout moment dans les divers compartiments, ainsi que les lois qui régissent leur comportement, et le plan d'action qu'on souhaite réaliser, l'ordinateur n'a aucune peine à élaborer les ordres nécessaires en fonction des résultats des tests pratiqués à des endroits choisis, résultats que, bien entendu, on lui communique aussitôt. Ces ordres conduisent à ouvrir ou fermer des dérivations, ou des vannes, ou des brûleurs, ou des jets de réactifs. On a pu ainsi automatiser de façon vraiment complète d'immenses raffineries, en dépit de la grande hétérogénéité des pétroles qu'on leur donne à traiter et de l'extraordinaire diversification des produits extraits.

Les capteurs et les actuateurs

Si vous avez l'œil perçant, vous aurez peut-être observé que l'emploi de l'ordinateur « en temps réel », que ce soit dans les applications militaires ou industrielles, suppose quelque chose de plus que le programme auquel vous a habitués l'emploi en « temps différé ». Lorsqu'un ordinateur est plongé dans le flot des événements, on ne peut pas lui fournir ses données d'avance, en même temps que son algorithme de calcul. Car ces données ne sont pas prévisibles, elles émanent du processus même. Il faut donc les prélever *sur le théâtre des opérations*, à mesure que celles-ci se déroulent. A l'autre bout, et pour la même raison, il ne suffit pas de lire sur un bulletin imprimé les résultats des cogitations *ordino*, il faut encore que ceux-ci se traduisent *dans l'instant et in situ* en rotations de vannes, déplacements de tapis roulants, traitements thermiques ou toutes autres actions nécessaires à la marche du processus. Les organes qui assurent la première fonction s'appellent des « **capteurs** » et ceux qui assurent la seconde s'appellent des « **actuateurs** ».

Comme fils d'Ève, vous ne devez avoir aucune difficulté à comprendre le travail de ce genre d'organes, car ils ont leurs équivalents exacts dans votre corps. Tous nos organes sensoriels sont des **capteurs.** Ils reçoivent les influx de divers types — lumineux, sonores, gustatifs, mécaniques, etc. — issus des objets et, dans des organes appropriés — rétine, tympan, papilles, peau, etc. — les transforment en influx nerveux qui vont informer notre cerveau. De même, dans l'industrie, le **capteur** de la rotation d'un arbre, par exemple, mesure l'angle dont l'arbre a tourné et transforme la valeur trouvée en un signal électrique ou un train de signaux. Dans certains cas, ces signaux, avant d'être envoyés à l'ordinateur, doivent être codés au moyen d'appareils supplémentaires. Mais de plus en plus l'industrie fournit des capteurs *incluant* le codage binaire, qui débitent donc directement des tops-tips. Devant la demande dévorante de l'ordinateur, la technique a développé des capteurs d'une astuce prodigieuse, mettant en œuvre une palette extraordinaire de phénomènes physiques, et capables de capter, pour les exprimer en tops-tips, toutes les grandeurs du monde et de la science. D'abord bien sûr les grandeurs simples telles que distances, sons et poids, mais aussi la concen-

tration d'un composant dans un liquide, la fréquence de vibrations d'une tige, les grincements de tels ou tels ressorts, le déplacement d'un corps fugitif dans la nuit, un « suspense » dans le chant des oiseaux, le reflet des yeux d'un chat, le changement d'une teinte, etc. En fait, on peut dire que la technique des capteurs est capable d'appréhender et traduire en signaux électriques codés une plus grande variété de phénomènes que nos cinq sens, encore que ceux qu'appréhendent nos sens le sont avec plus de finesse.

Quant aux **actuateurs**, ce sont évidemment des généralisations de nos mains. Ils ont d'ailleurs souvent l'aspect extérieur de bras, doigts, crochets, leviers qui poussent un ressort, ferment un chemin, ouvrent un clapet, tournent une roulette. Des signaux, qui cette fois *provoquent des actions* au lieu d'être, comme dans les capteurs, *induits par les phénomènes*, instruisent les actuateurs à accomplir les gestes voulus, comme notre main agit sous l'effet d'ordres issus de notre cerveau. On peut dire que, à l'inverse des capteurs qui transforment des phénomènes en signaux codés, les actuateurs transforment des signaux codés en phénomènes. Une illustration illuminante et familière de cette dualité est fournie par le binôme microphone — haut parleur : le premier est un capteur qui capte le son et le transforme en courant électrique, le second est un actuateur qui reçoit du courant électrique et le transforme en son.

Avec la floraison des capteurs enrichissant ses entrées et des actuateurs prolongeant ses sorties, l'ordinateur prend une dimension nouvelle puisqu'il peut directement interpréter le réel et agir sur lui après élaboration. C'est à partir du moment où il est doté de tels organes sensoriels et de tels organes d'action autour de son cerveau que l'ordinateur est entré dans les cauchemars sous la forme du « ***robot*** ». Nous allons voir dans le dernier chapitre si ces cauchemars sont justifiés.

On a vu l'ordinateur traiter de l'information pendant le cours même des événements — ce qu'on appelle en *temps réel,* par opposition au traitement après coup ou avant coup, qu'on appelle en *temps différé.* Vous avez trouvé des exemples dans l'interception de mobiles en mouvement très rapide, dans la dissémination, présentation et analyse des renseignements recueillis par le radar, dans la composition d'imprimerie, dans l'industrie, où il fallut développer des *capteurs* capables de transformer la mesure de toutes sortes de phénomènes en signaux électriques codés et des *actuateurs* capables de transformer des signaux électriques codés en actions sur toutes sortes de phénomènes. C'est un ordinateur précédé de capteurs et suivi d'actuateurs qu'on appelle un *« robot ».*

12

Médecine et enseignement

Abaissement ou élévation du médecin?

Tout le monde a lu que le médecin humain est périmé. Bientôt les malades seront soignés par des ordinateurs.

Soulignons tout de suite combien l'énoncé qui précède est trompeur. Au pire pourrait-on dire que les malades seront soignés selon des programmes conçus par des médecins. Des programmes pour diriger les examens, proposer des diagnostics et suggérer une thérapeutique, et qui seront évidemment rédigés d'après les préceptes les plus raffinés de la science médicale. Regardons cela de plus près.

Il y a d'abord une application des ordinateurs au soin des malades qui est évidente et ne renverse aucune notion reçue, c'est la gestion des hôpitaux ou cliniques. Cela comprend bien sûr la paie du personnel, ce qui peut paraître trivial, mais aussi la surveillance des stocks de médicaments en fonction des besoins prévus des malades présents, ce qui participe déjà aux soins proprement dits. On peut aller plus loin et faire confluer vers l'ordinateur une série de données qu'on relève sur les malades de façon routinière, comme la température quotidienne que rien ne nous empêche de récolter automatiquement sous forme de tops-tips à la sortie des thermomètres agencés en capteurs. Cela soulagera grandement les infirmières et permettra des pré-inspections très rapides à distance, où les cas d'alerte se détacheront au premier regard. D'autre part, la masse souvent considérable d'observations cliniques et de mesures de laboratoires qui concernent un malade peut être avantageusement codée en tops-tips sur des cartes perforées ou d'autres supports de mémoires. On constitue ainsi le dossier d'un malade sous une forme *ordino*, ce qui permet des opérations du plus haut intérêt. Ainsi un médecin peut demander à un *corpus* comprenant tous les dossiers *ordino* de tous les malades quel pourcentage des malades mâles de plus de 50 ans qui ont eu la maladie X pendant plus de Z jours ont présenté la séquelle Y, et ce chiffre éclairera puissamment ses réflexions. On collecte, classe et ventile les symptômes sur mille fois plus de malades qu'aucun médecin n'en verra jamais dans toute sa vie, et on peut former cent fois plus de corrélations que toutes les facultés du monde en dix ans. C'est par exemple à des statistiques de ce genre, faites sur des millions de cas avec tous les sous-classements désirables, par âge, sexe, profession, maladies connexes, taux de consommation, etc., qu'on doit la certitude actuelle de l'effet cancérigène de la fumée de cigarette. Quant à l'identité des malades, il est très facile de la maintenir secrète, si les malades le désirent, en désignant ceux-ci par des numéros qui ne sont décodables en noms propres que dans des fichiers et par des

clés que possède le seul médecin traitant. Remarquons que la mise sur ordinateur des dossiers médicaux pose des problèmes d'informatique extraordinairement lourds, car si on veut « décrire » un malade assez complètement pour que ça serve à la science, on constate qu'il faut consigner des centaines de paramètres, et la science justement n'est pas assez développée pour dire où on peut s'arrêter. Rien ne prouve qu'une chute faite pendant l'adolescence n'influe pas sur le diabète à 60 ans... En outre, il faut supposer que l'examen clinique d'où dérive la description du cas a été bien fait, ce qui souvent n'est pas sûr. Le problème du dossier médical informatisé préoccupe aujourd'hui de nombreuses équipes de médecins à travers le monde.

Allons d'un pas plus loin. Qu'est-ce que fait exactement un cardiologue quand il examine un électrocardiogramme? Il passe en revue dans son esprit tous les tracés normaux ou pathologiques présentant des allures voisines de pointes, descentes, courbures, périodes, et « reconnaît » par comparaison à quelle catégorie appartient celui qui lui est soumis. L'équivalent de ce processus mental peut évidemment se mettre en programme. Un appareil peut analyser un tracé qui occupe 2 mètres de papier en une série d'ordonnées très tassées, de manière à ne perdre aucune inflexion, mettons 5 par millimètre, plus que n'en résout l'œil du meilleur médecin. Chaque ordonnée est mesurée avec une précision égale à l'épaisseur du trait (aller au-delà serait illusoire), c'est-à-dire, mettons 1/2 millimètre. Si l'ordonnée la plus haute fait, mettons 3 centimètres, cela veut dire qu'on discerne 60 valeurs possibles dans chaque ordonnée, laquelle peut donc s'exprimer par un mot de 6 digits binaires ($2^6 = 64$). Un cardiogramme complet sera donc constitué par 5 000 mots de 6 digits... une misère. Comme on a isolé aujourd'hui 200 à 300 types possibles de cardiogrammes, un seul disque magnétique peut les contenir tous en mémoire. Et il n'y a pas de difficulté à concevoir un programme qui fouille systématiquement ce corpus et en extrait tous les modèles qui présentent une analogie assez marquée avec le cardiogramme soumis à examen (le traitement ressemble assez à celui du marieur qui cherche dans ses fichiers un conjoint assorti aux souhaits d'un candidat au mariage). On n'aura pas, bien sûr, une réponse unique, car en général le tracé examiné sera compatible avec plusieurs modèles, mais si l'ordinateur isole tous ceux-ci et imprime les affections correspondantes, on aura considérablement réduit le champ laissé aux supputations du médecin. On l'aura soulagé d'une besogne qui est en réalité peu intelligente et qu'un bon programme, incarnant le meilleur de la science actuelle, fait mieux que lui, plus complètement, plus sûrement et plus vite. Ce principe s'applique à d'autres relevés de laboratoire que les électrocardiogrammes, il couvre un vaste domaine qu'on pourrait appeler le *dépouillement automatique des mesures et analyses médicales.*

Nous voyons qu'un tel dépouillement se termine par un éventail de diagnostics possibles proposés au médecin. Cette aide de l'ordinateur peut aller plus loin. Si le médecin veut explorer systématiquement toutes les hypothèses retenues par le dépouillement automatique de l'électrocardiogramme, il confrontera chacune d'elles avec un *supplément* de données tirées des symptômes, de l'examen clinique, d'autres mesures de laboratoire, etc. Or cette activité est au fond de même nature que la précédente, encore que de plusieurs degrés plus complexe, car elle se ramène en fin de compte à rechercher, dans la totalité de la matière médicale, un ou plusieurs *archétypes*, préalablement isolés, connus et fichés, et qui concordent avec le cas examiné. Vous concevez donc que si toute la matière médicale était disponible en mémoire, sous forme codée et de façon qu'on puisse y entrer de multiples manières (à partir des symptômes, ou

des paramètres du malade, ou des profils pathologiques préalablement reconnus dans des domaines partiels), le médecin pourrait étendre considérablement ses consultations et se faire proposer l'ensemble des meilleures hypothèses compatibles avec le cas qu'il étudie. Il tirerait de l'ordinateur une *aide au diagnostic* d'une extraordinaire puissance, puisqu'il aurait parcouru tous les cas avec les yeux de toute la science et non pas les seuls cas que son expérience forcément partiellle présente à sa mémoire inévitablement défaillante et à ses connaissances nécessairement limitées.

Ce n'est encore bien sûr qu'un idéal qu'on est très loin d'atteindre aujourd'hui, parce que la matière médicale est si fantastiquement ramifiée et embrouillée, et que la science médicale est encore si incertaine, que nul ne peut dire par quels chemins on est sûr de l'explorer en entier. Beaucoup prétendent que cet idéal est par nature inaccessible parce que le rapport médecin-malade comporterait une série d'éléments qui ne seront jamais chiffrables et ressortissent d'une part à l'intuition ou à l'autorité du médecin et d'autre part au psychisme du malade. Que veulent dire exactement ces tenants du « flair » contre la « science »? J'entre, dira l'un d'eux, dans la chambre du malade. Déjà la traversée de son appartement, les mines de ses familiers, le décor et l'atmosphère autour de lui m'auront apporté des effluves nombreux. Puis je vois son œil éteint, ses lèvres exsangues, ses joues tombantes, j'entends sa voix douteuse m'interroger, et j'ai en quelques instants gravé en moi une partie du tableau clinique qui est essentielle pour bien juger, et dont aucun ordinateur ne saura jamais tenir compte comme moi. C'est l'irremplaçable « contact humain » qui a joué selon ses voies directes et mystérieuses.

Les « mystères du contact humain » sont invoqués souvent à tort et à travers dans le seul but, inavoué, de préserver des habitudes que le regard de la science menace de déranger. Si cependant on doit reconnaître que le médecin perçoit en effet, dans la réalité vécue par son client, une foule de signes importants, est-il vrai qu'aucun calculateur ne saura jamais les exploiter? Bien qu'ils aient l'air impalpables, ces signes pourraient certainement être exprimés dans un langage standardisé avec des paramètres quantifiables. Mais il y faudrait des analyses extraordinairement méticuleuses qui demanderaient des décennies à des équipes considérables. En outre, une fois ce travail fait, et un tableau humain, (tel que celui que nous avons esquissé tout à l'heure) traduit par quelque chose comme, mettons, cinq mille mots binaires de 32 digits, il faudrait encore qu'on intègre cette information dans l'ensemble des données plus biologiques qui décrivent le malade et son mal, et qu'on sache s'en servir objectivement. Un programme à cet effet durerait, sur le plus puissant ordinateur du monde, beaucoup plus de temps que n'en met le médecin à tout voir et comprendre d'un coup d'œil.

On rencontre ici quelque chose qu'il est important que vous sachiez. En dépit de sa fantastique vitesse, il y a des domaines où *l'ordinateur reste plus lent que l'homme.* L'homme possède en effet un étonnant pouvoir pour reconnaître et appréhender des configurations complexes comme des messages uniques — par exemple pour distinguer un cercle d'une ellipse — ou pour intégrer les touches qui composent un panorama. Pour qu'un ordinateur « comprenne » ce que l'homme apprend par ce genre de flashes, il doit en analyser le contenu grain par grain et pas à pas, et il a beau marcher à la microseconde, s'il lui faut dix millions de pas pour tout explorer, il mettra dix secondes, et sera battu par l'homme qui aura tout saisi entre deux clignements de cils. Certes le coup d'œil du médecin embrasse aussi une multitude de données *inutiles*,

mais son cerveau possède cette autre aptitude de s'en débarrasser très vite. L'ordinateur pourrait peut-être regagner la compétition s'il ne traitait que les éléments saillants de la situation, mais il faudrait qu'on pût le programmer pour qu'il sache les dégager d'emblée, et cela à son tour supposerait que l'homme ait su les isoler. Or nous avons vu que la science médicale n'y est pas encore parvenue, ce qui ne laisse effectivement place pour l'instant qu'à une intuition éclairée.

Il est donc vrai qu'à l'heure actuelle on ne peut pas encore mettre en ordinateur tous les paragraphes de tous les traités, tous les chiffres de tous les laboratoires, toutes les mines de tous les malades. Les modernistes sensés en conviennent. Ce n'est pas une raison pour ne pas y mettre dès aujourd'hui tout ce qui est connaissable et codable aujourd'hui. Cela ne peut qu'augmenter la puissance du médecin, donc aussi sa responsabilité. C'est entendu, le médecin de chair sera là pour voir la chair, mais en dehors de cette tâche combien d'autres, ressortissant aux sphères de la mesure et du dossier, ne pourra-t-il confier à l'ordinateur avec grand profit pour tous. S'en priver aujourd'hui, c'est un peu comme si en 1750 on avait interdit le microscope au botaniste. Loin de diminuer le praticien, l'ordinateur lui permet (en principe) de mobiliser entre ses mains, au service de son patient, la totalité des connaissances disponibles et des observations quantifiables, c'est-à-dire en fait d'appeler à lui en consultation les ombres de milliers de médecins... en tout cas de ceux qui prennent des notes. Et Dieu sait que le praticien doit être calé, et donner le meilleur de lui-même, pour pouvoir tirer parti d'une consultation aussi étendue. Il n'est jamais plus complètement, plus noblement homme que quand l'ordinateur lui a mis la science en mains et lui a dit « te voilà averti, à toi de jouer ». Il ne devient pas un sous-médecin, il devient tous les médecins.

Pour ma part, je crois que l'informatique médicale devrait être poussée avec le maximum de moyens. C'est le plus grand espoir actuellement offert aux hommes d'améliorer leur santé.

L'ordinateur dans l'enseignement

Voici encore un domaine où il semble curieux que l'ordinateur puisse pénétrer. Voyons comment il y prétend, et s'il y a réussi.

On comprend sans peine certaines applications. Par exemple, un ordinateur dans une école peut être facilement programmé pour tenir ce qu'on peut appeler la « gestion pédagogique » des élèves. Cela consiste d'abord à mettre en mémoire des dossiers individuels où sont inscrits, en code naturellement, les présences et absences de chaque élève, les cours qu'il a suivis, les devoirs qu'il a faits, les notes qu'il a obtenues, les observations des professeurs et les siennes propres. Jusqu'ici ces données voltigeaient et s'évanouissaient comme des vapeurs. Si désormais on peut en disposer d'une façon exhaustive, et les consulter à la fois rapidement et sélectivement, on dispose d'un outil remarquable pour diriger les études. On peut en effet suivre beaucoup mieux, et à la semaine, l'évolution intellectuelle des élèves, y dégager les points faibles et forts, constituer des sous-groupes homogènes pour des travaux supplémentaires ou des discussions collectives, et même étudier sur des statistiques désormais valables les corrélations qui peuvent exister entre les aptitudes dans diverses disciplines, entre le succès et le travail, entre les résultats et les méthodes d'enseignement, etc.

Une autre application claire est l'emploi de l'ordinateur, surtout lorsqu'il est terminé par une console de visualisation, pour aider l'étudiant en expédiant à sa place, rapidement, des tâches annexes qui ne lui apprennent rien et freinent son avance. Par exemple, une fois qu'un garçon a bien assimilé son arithmétique, il ne sert plus à rien, lorsqu'on veut lui enseigner l'algorithme permettant de résoudre une équation, de l'obliger à calculer lui-même les valeurs numériques. Il peut alors avec profit demander à l'ordinateur de lui livrer ces valeurs et concentrer son effort mental sur l'algorithme de résolution. Il peut aussi, s'il étudie des structures grammaticales, ou des réactions chimiques, ou le relief de l'Iran, se faire afficher des phrases, ou des formules, ou des cartes, et poser des questions à l'ordinateur en mode conversationnel. L'ordinateur répond du tac au tac en renvoyant des phrases modifiées, ou l'image d'un précipité, ou de nouvelles lignes de niveau sur un terrain. Il faut pour cela avoir mis dans l'ordinateur des programmes assez savants, prévoyant toutes les questions que les élèves pourront poser dans toutes les situations et les réponses à y formuler. Ce système est très fécond mais terriblement coûteux, d'abord parce qu'il faut confectionner les dits programmes qui sont très lourds et très complexes, ensuite parce qu'un seul élève peut bénéficier à la fois de la console, et qu'une heure de fonctionnement d'un ordinateur peut valoir 3 000 F. Aussi cette assistance à l'étude individuelle où l'ordinateur joue le rôle du partenaire donnant la réplique dans une répétition théâtrale n'a-t-elle pu entrer en pratique que dans les études supérieures ayant une valeur élevée pour la société. C'est le cas par exemple pour la médecine. On trouve aujourd'hui dans plusieurs facultés des installations très sophistiquées où un étudiant essaye son savoir devant une console. Il voit d'abord affichée la description d'un malade avec des symptômes et des mesures de laboratoire, plus éventuellement un électrocardiogramme ou une photo de poumon. Il formule alors un diagnostic qu'il communique par son clavier à la machine. Celle-ci répond si elle l'accepte ou non en première hypothèse. Et le dialogue s'engage, comme si le malade était tapi en morceaux dans le ventre de l'ordinateur et répondait par sa voix aux questions, épreuves et sondages de son médecin. L'étudiant demande des analyses, et aussitôt s'affichent des résultats tirés d'un des cas cliniques réels qui avaient été déposés en mémoire. S'il ne demande pas assez d'examens, la machine proteste, car en même temps qu'elle simule le malade, elle remplace aussi le professeur. Puis l'étudiant formule un traitement. Si la machine est d'accord, il reçoit 20 sur 20 et l'exercice s'arrête. Sinon l'écran se couvre d'écritures proposant d'autres soins. L'étudiant proteste et dit à la machine : « Pourquoi me suggères-tu tel médicament à la place du mien ? » Et la machine répond sur l'écran : « Parce que ton malade est un hépatique de 72 ans. Regarde les contre-indications de ton médicament. » Et ainsi de suite. La machine joue en plus un rôle d'espion, car elle imprime toutes les démarches de l'étudiant afin que son professeur — le vrai — puisse y réfléchir.

Voilà déjà, pensez-vous, pas mal de services précieux que l'ordinateur peut rendre à l'enseignement. Mais ce n'est encore rien à côté de l'ambition qui s'est emparée, voici 5 à 6 ans, des chercheurs et des grandes compagnies d'informatique. Pour bien la comprendre il faut faire une rapide incursion dans ce qu'on appelle « L'Instruction Programmée ». Elle est surtout destinée à l'étude individuelle, sans professeur, et présente la matière à étudier sous forme de séquences de petits morceaux appelés « items » en anglais, enchaînés, dont chacun contient une question, une sorte de micro-exercice que l'élève doit réussir pour passer à l'item suivant. S'il répond mal,

il est envoyé à un autre développement. Ces séquences ressemblent vaguement à un programme d'ordinateur, d'où le nom de la méthode. Si l'élève a bien répondu et le constate sur-le-champ, il éprouve ce qu'on appelle un « renforcement », qui est le moteur essentiel de l'apprentissage.

Il faut donc qu'il réponde bien le plus souvent possible. Pour s'en assurer, et vu qu'on n'en sait pas assez aujourd'hui sur les mécanismes mentaux de l'ignorant qui cherche à apprendre pour pouvoir *prévoir* son comportement devant un item, on est obligé d'*expérimenter*. On propose des ébauches du programme à des cobayes, on teste et interprète leurs réactions, et on corrige et re-corrige l'ébauche jusqu'à ce que 90 % des cobayes réussissent 90 % des items. Ces essais préalables d'un cours programmé constituent en fait la vertu essentielle de la méthode, ils introduisent enfin dans la pédagogie une expérimentation systématique qui permet de beaucoup mieux adapter le cours aux besoins de l'élève que ne peut le faire un professeur sur la base de ses intuitions et présomptions. Seulement, le cours qui sort de ces réfections empiriques, s'il se situe plus près des démarches moyennes des élèves, ne se situe tout de même que le long d'une *moyenne*. Une fois qu'il est imprimé *ne varietur*, *tous* les élèves qui l'emploieront devront s'accommoder des *mêmes* séquences. L'idéal serait qu'on pût franchir un pas de plus vers le « cours sur mesure » en préparant une multiplicité de chemins possibles et proposant à chaque carrefour à l'élève celui qui tient compte au mieux de ses idiosyncrasies personnelles, et qui peut être différent de celui qui convient à son congénère. Chaque potache aurait alors l'équivalent d'un précepteur privé le conduisant par la main.

Nous approchons de l'ordinateur. Que faut-il pour parvenir à cet enseignement tout à fait « *adaptatif* » ? Une chose hélas aussi simple à définir que difficile à réaliser. Il faut comprendre *la cause de l'erreur* commise par l'élève à un certain item, car ce n'est qu'alors qu'on pourra l'aiguiller correctement vers celle des multiples bretelles qui a été prévue pour traiter ladite cause.

C'est ce travail de *diagnostic de troubles intellectuels* qu'on espérait pouvoir faire exécuter par l'ordinateur assez vite pour que l'élève reçoive ses consignes dans les quelques secondes qui suivent l'émission de sa réponse. On espérait que l'ordinateur pourrait parcourir toutes les filières qu'avait suivies l'élève, et qui auraient été conservées en mémoire, et par une série de tests très subtils sur les erreurs et succès passés, ainsi que sur le lien logique entre la question posée et la réponse émise, se faire une idée de la nature et de l'origine des incompréhensions qui ont dévoyé l'élève, afin de guider son prochain pas en conséquence. Des efforts considérables ont été déployés pour arriver à cette détection automatique des causes d'erreurs. L'enjeu était énorme, aussi bien sur le plan culturel que sur le plan commercial, car si on résout ce problème, on ouvre le champ à un mode d'enseignement d'une puissance inouïe, et aussi à un marché inépuisable si on songe au nombre de centaines de millions d'écoliers qui demanderont des ordinateurs.

Malheureusement, cet espoir a été déçu, et c'était à prévoir. Le processus par lequel un cerveau se meuble de connaissances et d'aptitudes nouvelles est si complexe, et encore si mystérieux, qu'il n'y avait aucune chance qu'on apprît en quelques années à détecter quels mécanismes ont engendré tel ou tel comportement de l'élève. Il y faudra une masse de recherches immensément plus grande. Or, dès lors que l'homme ne sait pas pourquoi un élève s'est trompé (ou a réussi), l'ordinateur ne le saura pas davantage puisqu'il attend de l'homme qu'il lui dise comment acquérir ce savoir.

C'est là une leçon importante à glaner dans notre promenade, et dont nous apprécierons tout le poids au dernier chapitre : l'ordinateur n'exécute que les traitements dont le programmeur peut tracer la voie. Si les meilleurs maîtres ne savent pas pourquoi tant de gosses qui font bien $7 + 5 = 12$ et $7 + 6 = 13$, et $7 + 8 = 15$, se bloquent tout à coup devant $7 + 4 = ?$, aucun ordinateur ne pourra le découvrir à leur place. Du moins en l'état présent des choses. C'est pourquoi, pendant longtemps encore, l'ordinateur ne pourra pas réaliser le rêve d'un enseignement individuel adaptatif, et ne fournira en pédagogie que des services auxiliaires tels que : gestion du travail scolaire, exécution de corvées intellectuelles, présentation de documentations complémentaires textuelles, visuelles ou sonores. C'est déjà beaucoup, et ça deviendra décisif quand ces services seront moins chers.

On a dissipé les malentendus sur l'usage de l'ordinateur en médecine en montrant qu'il conduit à placer auprès du malade non plus aucun médecin, mais au contraire tous les médecins. Et l'on a situé plus exactement les services qu'on peut tirer de l'ordinateur dans l'enseignement.

13

Applications scientifiques

Le mystère des gènes

Je termine cette revue des applications de l'ordinateur par celles qui ont présidé à sa naissance et qu'on appelle « scientifiques », par opposition aux applications à la « gestion ». Il est essentiel de bien les comprendre, car il règne à leur sujet des contresens graves.

N'importe qui pourrait être riche. La géologie est en effet assez avancée pour pouvoir dire avec certitude que dans telle région de 10 000 km² de l'Alaska il y a de l'or. On sait d'autre part très bien comment il faut tamiser les eaux, ou souffler les boues, ou tailler les roches, pour tirer l'or de là où il existe. Il ne reste dès lors qu'à explorer systématiquement tous les ruisseaux, tous les étangs, toutes les falaises du territoire en question, selon un parcours mathématiquement déterminé pour passer partout, partout, et on est sûr de tomber sur un filon. Le programme de prospection peut même inclure des tests sur la qualité probable des filons, car la géologie et la chimie permettent de l'apprécier, de sorte qu'on n'ouvrira une mine qu'à bon escient.

Pourquoi les gens ne font-ils pas cela? Vous l'avez sûrement deviné : c'est pour la raison simple mais dirimante que l'exécution effective de ce programme par un homme ou même une société demanderait de nombreux siècles. Eh bien, il en va de même dans les problèmes scientifiques où on fait intervenir l'ordinateur.

Prenons par exemple la structure des molécules. Vous savez que la molécule d'un composé chimique contenant divers éléments est constituée par les atomes de ces éléments assemblés d'une certaine manière. De quelle manière? Il importe de connaître cette architecture intime, car beaucoup de propriétés de la substance chimique en dépendent : son affinité pour une autre substance, sa solubilité dans l'eau, sa conductibilité électrique, son magnétisme, sa couleur (c'est-à-dire les fractions du spectre lumineux qu'il absorbe ou qu'il réfléchit), etc. Or, après un siècle et demi de recherches et spéculations, les chimistes ont déterminé toutes les lois selon lesquelles les atomes s'attirent ou se repoussent. Ils peuvent donc écrire les équations mathématiques régissant leurs coordonnées, et les solutions de ces équations donnent les positions que doivent occuper les atomes pour former la configuration équilibrée de la molécule. Les mathématiciens, d'autre part, savent écrire l'algorithme de tous les calculs qu'il faudrait faire pour résoudre ces équations. Mais la lourdeur de cet algorithme croît exponentiellement avec la complexité de la structure, de sorte qu'il n'a pu être conduit jusqu'au bout que pour les molécules les plus simples, comprenant au plus quatre atomes eux-mêmes très simples. Pour une molécule moyenne de

20 atomes en tous genres, le calcul à la main de l'architecture connaissant les atomes composants demanderait des années, et des siècles pour une grosse molécule comme celle des substances organiques qui comprend souvent 100 atomes. Nous sommes alors de retour à la situation du chercheur d'or : nous avons toute la science nécessaire pour, *en principe*, réussir notre quête, mais *en pratique*, au rythme *homo*, le chemin pour y parvenir est beaucoup trop long. Alors on renonce.

Du moins, on renonçait jusqu'à l'avènement de l'ordinateur. Car celui-ci change tout. Il peut emmagasiner l'algorithme en question sous forme de *programme*, et l'identité des atomes composants sous forme de *données*. Il calcule alors les positions des atomes en des durées qui, pour les molécules organiques les plus complexes connues à ce jour, ne dépassent pas quelques heures (vous voyez que même à la vitesse *ordino*, c'est encore une durée non négligeable). Et on peut même envoyer les résultats sur une console de visualisation à tube cathodique dotée d'un programme d'édition tel que l'édifice se dessine tout seul sur l'écran avec l'apparence du relief, chaque genre d'atomes étant représenté par un petit symbole adéquat et les divers liens entre atomes par des traits pleins ou pointillés. C'est ainsi qu'on a déterminé l'architecture tourmentée des molécules du fameux acide nucléique ADN, composant de base de nos gènes, d'où est sortie toute la biologie génétique moderne, car c'est à partir de cette architecture qu'on a pu déchiffrer quelques-uns des plus profonds mystères de la matière vivante. Les résultats du calcul ont d'ailleurs été confirmés ensuite dans tous leurs détails par l'observation directe au microscope électronique et par toutes sortes d'inférences indirectes tirées de l'expérience.

Qu'est-ce au juste qu'une équation?

Le concept d'*équation*, que j'ai invoqué tout à l'heure, est souvent mal compris. On l'assimile, sans trop y regarder de près, aux concepts de « *formule* » ou d' « *expression* » mathématiques. Or, il importe de saisir la différence si on veut comprendre le rôle de l'ordinateur dans les sciences.

Entre la pression p d'une masse donnée de gaz, sa température T et le volume v qu'il occupe, règne une certaine relation qui s'écrit :

$$p = \frac{RT}{v - b} - \frac{Q}{(v - b)^3}$$

où R est une constante universelle tandis que Q et b sont des constantes dépendant du gaz. Cette égalité est une *formule* lorsqu'on y considère toutes les grandeurs comme *connues*. Mais cette même égalité devient une *équation* si on demande quel volume v il faut offrir à un gaz donné pour qu'à une température T donnée il prenne une pression p qu'on s'impose. On dit alors que l'égalité est une « équation en v » pour donner à entendre que la lettre v désigne maintenant une *inconnue* et que l'égalité exprime non plus un état de fait, mais une *condition* qu'on impose à l'inconnue de satisfaire, et qui devra permettre de la déterminer au moyen de « déductions à rebours » du genre de celles qu'on fait pour retrouver un objet à partir de ses propriétés. C'est ainsi qu'en édictant qu'il épouserait celle qui chausserait l'escarpin trouvé au bal, le prince écrivait une « équation » pour le pied inconnu, équation que seule, comme vous savez, Cendrillon put satisfaire.

Dans notre exemple du gaz, l'inconnue était simplement une *grandeur*, le volume v, représentable par une seule lettre. Mais dans des problèmes plus raffinés c'est toute une *fonction* qui peut être l'inconnue à déterminer, c'est-à-dire une *relation* entre deux grandeurs (ou plusieurs). Ainsi, en météorologie, on cherchera la fonction qui lie la vitesse du vent à l'endroit du globe dans une atmosphère donnée. Les équations qu'il faut écrire pour déterminer des *fonctions* — et qu'on distingue, quand c'est nécessaire, par l'adjectif « fonctionnelles » — sont naturellement plus compliquées que celles qui déterminent des *grandeurs*. On rencontre aussi, à un étage plus haut, des *systèmes* de plusieurs équations demandant qu'on détermine à la fois plusieurs inconnues liées entre elles, inconnues pouvant être, soit des « grandeurs », soit des « fonctions ». La plus simple des équations concevables est celle dite « du premier degré », qui impose à une grandeur inconnue z de satisfaire une égalité du type

$$az + b = 0$$

où a et b sont des coefficients. Il est patent que n'importe quelle valeur de z, multipliée par a et ajoutée à b, ne donne *pas*, tout justement 0. L'égalité écrite n'est donc *pas* vraie au départ, et ne prétend pas l'être; elle constitue une *condition* imposée à z, qui ne sera satisfaite que pour une certaine valeur de z. En l'espèce, cette valeur est immédiate. Elle est donnée par

$$z = -\frac{b}{a}$$

et s'appelle la « solution » (ou encore la « racine ») de l'équation. Cendrillon était la solution de « l'équation au pied » du prince.

Une équation ressemble beaucoup à une devinette où vous devez trouver à quel animal je pense. Je vous dis par exemple : cet animal, *si* je lui marche sur la queue, aboit : quel est-il? Une équation mathématique formule de la même manière des *indices* sur l'inconnue à déterminer : si, sur ces inconnues, j'effectue telles opérations, j'obtiens tel résultat; que valent les inconnues? Et la démarche pour résoudre une équation ressemble, logiquement du moins, à la recherche d'un criminel à partir des indices qu'il a laissés.

Si vous avez bien compris ces notions, vous êtes prêt pour recevoir deux grands secrets de la science que le public ne connaît pas assez. Le premier secret, c'est que l'étude de la nature ne peut aller que rarement jusqu'aux valeurs mêmes des grandeurs qu'elle voudrait cerner, ou jusqu'aux lois toutes explicitées qui lient les grandeurs entre elles. A l'instar des énigmes policières, la nature ne nous livre, sur ces grandeurs, ou sur ces lois, que des indices *indirects*, sous la forme de *conditions* auxquelles elles satisfont. Quand on écrit mathématiquement ces conditions, on obtient, précisément, des *équations*, qu'il faut ensuite résoudre pour remonter aux coupables. Le second secret, qui vous surprendra davantage, c'est que la mathématique *permet d'écrire beaucoup plus d'équations qu'elle n'en sait résoudre*, si on entend par « résoudre » une équation le fait d'exprimer l'inconnue par une *formule* explicite, bâtie avec des fonctions *connues* des coefficients, paramètres ou autres variables qui figuraient dans l'équation. Par exemple $z = \frac{b}{a}$ était la solution de l'équation $az + b = 0$ *exprimée par une formule*. Dès que les équations écrites se compliquent un peu, on ne peut plus les résoudre par des formules. On doit se rabattre sur une résolution appelée « point

par point », qui n'est jamais qu'*approchée*, c'est-à-dire qui donne une liste des valeurs *approximatives* des solutions pour chaque groupe de valeurs des données. Et il faut mettre en œuvre tout un algorithme, dit « de résolution », pour calculer ces valeurs séparées dont l'ensemble forme « la » solution lorsqu'elle n'est pas exprimable par une formule close.

Ces deux secrets vont vous éclairer le drame des physiciens qui étudiaient la nature, et le dénouement qu'en donne l'ordinateur. Ce drame tient en peu de mots : *les physiciens étaient presque toujours conduits à des équations non résolubles par des formules.* Pire. Des équations si lourdes qu'ils ne se souciaient même pas d'en dresser les algorithmes de résolution « point par point », car ils savaient d'avance que ces algorithmes seraient inexécutables en pratique, comme l'algorithme de la recherche sans omission du filon d'or. Alors ils échafaudaient des « *théories* », dont le but principal était de *simplifier les équations.* Pour y parvenir, ils faisaient d'ingénieuses hypothèses permettant de négliger l'effet de telle grandeur, ou de schématiser l'incidence de telle autre, ne retenant, après avoir beaucoup pétri les phénomènes, que leurs traits saillants et quintessenciés, jusqu'à ce que ceux-ci se laissent décrire par des équations assez simples pour qu'un homme puisse les résoudre dans des délais acceptables. On pouvait dire, en caricaturant un peu, que tout le travail du théoricien consistait à « ramener les phénomènes au second degré », car l'équation qui a la forme d'un polynôme du second degré égalé à zéro, c'est-à-dire la forme $ax^2 + bx + c = 0$, est la seule qu'on sache résoudre à vue par une formule simple — c'est pourquoi d'ailleurs elle accapare le bachot. Inutile de dire que toutes ces hypothèses n'ayant d'autre but que de simplifier les équations faisaient perdre une grande partie du réel et donnaient lieu à mille controverses.

L'ordinateur a renversé ces anciennes bornes de la pensée. On peut désormais écrire sur les phénomènes *tout* ce qu'on en sait, sans rien négliger et sans s'effrayer de la complexité des équations auxquelles on parvient, sachant que l'ordinateur pourra quand même exécuter les calculs requis pour les résoudre. Bien sûr, il faut pour cela que des analystes découvrent les algorithmes de résolution, et c'est souvent un problème majeur. Il a fallu mobiliser à ce travail les meilleurs cerveaux mathématiques et développer une discipline aujourd'hui considérable qu'on appelle « *mathématique pratique* » ou « *mathématique appliquée* » ou « *mathématique numérique* » (sachez entre parenthèses que cette discipline n'emprunte rien aux fameuses « mathématiques modernes »). Mais on y est parvenu. On connaît aujourd'hui les algorithmes de résolution « point par point » de presque toutes les équations qu'on peut être amené à écrire. Ces algorithmes ont presque toujours la nature de ces rondes d'*approximations successives* dont je vous ai parlé dans la première partie. On donne aux paramètres un groupe de valeurs numériques et on essaye une première valeur d'inconnue qui satisfait l'équation ainsi numérisée *à peu près*, laissant une certaine erreur. De là on déduit *à bon escient* une seconde approche laissant une erreur moindre, puis une troisième, quatrième... centième... jusqu'à ce que l'erreur soit descendue au-dessous d'une tolérance admise. Puis on recommence avec un autre groupe de valeurs des données. On est sûr de pouvoir respecter toute tolérance désirée, aussi petite soit-elle, en répétant assez souvent ces itérations, dès l'instant où celles-ci « convergent » comme on dit, c'est-à-dire font à chaque coup diminuer l'erreur résiduelle. Tant pis si la convergence est lente, l'ordinateur précisément est là, avec sa supervitesse, pour parcourir, si besoin est, des centaines de cycles d'itérations. Ce qui n'empêche pas les

analystes de consacrer beaucoup d'efforts à découvrir l'algorithme le plus rapidement convergent, car au prix que coûte la seconde de travail sur un gros ordinateur moderne, le cycle d'itération vaut cher. Mais ce sont là des problèmes de spécialistes. Pour vous, il suffit qu'au terme de cette parenthèse vous ayez compris clairement les idées suivantes sur le rôle de l'ordinateur dans l'aventure scientifique :

— le *savant* doit savoir écrire les formules ou équations régissant les phénomènes étudiés;

— l'*analyste* doit pouvoir dresser un algorithme permettant de numériser ces formules ou résoudre ces équations;

— et l'*ordinateur* assure que cet algorithme, même gros, pourra être exécuté dans des délais acceptables.

L'instrument conditionne l'art

Est-ce là tout? pourriez-vous me dire, déçu. L'ordinateur n'est donc que le four qui cuit le gâteau de la science, sans aider celle-ci à le préparer?

Ici se place une réflexion capitale.

D'abord, ça n'est pas rien que de cuire le gâteau puisque, autrement, il resterait indigeste. Combien de chemins qu'on avait ouverts, mais qu'il fallait laisser en friche à cause de l'épaisseur des taillis rencontrés, découvrent enfin les terminus bouleversants où ils conduisaient! Je vous ai déjà cité l'architecture des molécules. Mais il y a cent autres domaines où on peut pour la première fois, grâce à l'ordinateur, aller jusqu'au bout du connaissable : le flot exact des électrons à travers des cristaux, ou celui du sang dans les veines, les raisons de la couleur des corps, les circonvolutions de la chaleur dans l'air, les lignes de résistance au fond des sols ou des bétons, le contournement des obstacles par les sons, bref tous les domaines où les lois physiques qu'on connaissait n'avaient jamais pu être exploitées complètement parce que ça demandait trop de calculs. Il n'y a en fait plus aucun laboratoire de recherche de quelque conséquence dans le monde qui n'ait son ordinateur, ou ses accès à un ordinateur. Et rendez-vous compte que de pouvoir désormais calculer tous ces festons délicats dans le jeu de la nature ne permet pas seulement de satisfaire la curiosité. Cela permet surtout de *nouvelles avances* vers des situations de plus en plus complexes qui, sans l'ordinateur, seraient restées impénétrables. Les transistors n'auraient pas connu leur extraordinaire développement si on n'avait pu écrire que les lois du mouvement des électrons, sans pouvoir en déduire pratiquement tous les remous qui peuvent se produire quand ils traversent des cristaux compliqués.

Ouvrons ici une parenthèse pour dire quelques mots d'un type d'ordinateurs que nous avons passé sous silence, ceux dits « analogiques ». Avec tous les ordinateurs du type traité dans cet ouvrage (et qu'on appelle « digitaux » ou « numériques » quand on veut les distinguer des « analogiques »), toutes les informations relatives à tous les phénomènes qu'on étudie ou qu'on contrôle, sont transformées en *nombres*, sur lesquels on fait des calculs programmés. Il existe au contraire des cas où les grandeurs physiques dont on traite, peuvent être représentées par des *courants* ou des *tensions* électriques, de telle façon que ce qui arrive aux grandeurs est exactement **analogue** à ce qui arrive à ces courants et tensions dans certains *circuits*. Ce qui s'exprime mathématiquement en disant que, dans ces circuits, les courants et tensions

obéissent aux *mêmes équations* que les grandeurs naturelles, de sorte que les dits circuits constituent des modèles des phénomènes étudiés.

Il n'y a plus dès lors besoin de calculer, il suffit de monter les circuits idoines, d'y laisser évoluer spontanément les courants et tensions, et d'observer les valeurs qu'ils prennent en fin de compte : ces valeurs représentent celles que prendront les grandeurs physiques étudiées dans leur environnement naturel. Et si on veut examiner comment ces valeurs varient lorsque tel ou tel paramètre de l'environnement se modifie, et bien on retouche les paramètres des circuits modèles, et on enregistre les nouvelles valeurs finales des courants et tensions. C'est un peu comme un jeu de poupées employé pour donner l'image d'un jeu d'hommes.

Un « calculateur analogique » n'est rien d'autre qu'un équipement comprenant des circuits électroniques modifiables et des instruments pour mesurer courants et tensions et pour les faire varier. En fait, il ne « calcule » rien, il laisse s'établir les courants et les tensions. Nous n'avons pas traité de ces engins pour deux raisons : d'abord, ils ne sont intelligibles qu'au moyen d'une dose d'électronique bien plus lourde que celle que vous avez dû absorber; ensuite, leur champ d'application reste étroit, car il n'y a que peu de phénomènes qui se déroulent dans la nature **comme** des courants et tensions dans des circuits.

Le bond en avant le plus sensationnel que la science et l'humanité ont pu faire, simplement parce que l'ordinateur a permis de pousser les calculs jusqu'au bout sans rien négliger, est évidemment la conquête de la Lune. Là encore, on connaissait complètement les lois qui régissent le mouvement des masses dans l'espace. On en connaissait le principe depuis Newton, et la mise en équation définitive de ces lois avait été réalisée dès le début du XIX^e^ siècle par Laplace dans son illustre « Traité de Mécanique Céleste ». Ces équations étant insolubles par des formules quand il y a beaucoup de corps célestes en présence, les astronomes et mathématiciens se mirent à faire des « théories », au sens que je vous ai décrit tout à l'heure, pour pouvoir obtenir quand même des résultats approchés dans des configurations particulières, comme celle des trois seuls corps Soleil, Terre et Lune. Ils y parvinrent en tendant à leurs extrêmes limites la patience des bureaux de calcul et les crédits publics, car pour dresser à la main les éphémérides de ce groupe de trois astres, avec les sept à huit décimales qu'on connaît aujourd'hui, il a fallu que, sur le budget de l'État, des milliers de dames triturent les tables de logarithmes dans des centaines de bureaux obscurs pendant des dizaines d'années.

Mais l'effort aurait dû encore être multiplié par mille si on avait voulu résoudre à la main les équations astronomiques pour la configuration formée par *quatre* corps dans l'espace : la Terre, la Lune, le Soleil *et* une *fusée*, dont le dernier pèse un million de fois moins que le plus léger des trois autres, de sorte que les forces agissant sur lui doivent être calculées avec beaucoup plus de précision.

Mais ce problème reste pleinement dans les moyens de l'ordinateur, qui peut même exécuter le calcul si vite que ses résultats sont disponibles en quelques secondes, de sorte qu'on peut les renouveler à mesure que la fusée vole, « en temps réel ». Et le plus merveilleux, le plus fantastique dans cette affaire, c'est que les calculs faits ainsi sur la base de lois formulées voici 300 ans, fournissent les positions des mobiles au mètre près, apportant à Newton une confirmation d'une plénitude qu'on n'a rencontrée jusqu'ici dans aucun autre secteur de la science. D'autres ordinateurs calculent, également « en temps réel », toutes les manœuvres que doivent accomplir les nombreux

appareils de bord tels que sources d'oxygène, thermostats, baromètres, etc. Des *capteurs* mesurent en permanence les conditions régnant à bord et transmettent leurs observations par radio à l'ordinateur qui en déduit des ordres adéquats qu'il transmet en retour par radio à des *actuateurs*. C'est ce qu'on appelle la « télécommande calculée ».

Vous voyez déjà, par cet exemple célèbre, combien il serait faux de dire que l'ordinateur n'apporte rien d'essentiel sous prétexte qu'il ne fait que mener rapidement à terme les calculs dont le savant a dû prescrire le but et la marche. C'est bien à l'homme que reste dévolue l'obligation de penser les phénomènes, mais il pense différemment selon les outils qu'il possède. Ceux-ci ont eu une telle influence sur son horizon intellectuel qu'on a pu parler de « l'âge du silex », de « l'âge du bronze » et de « l'âge du fer ». L'attelage du cheval par le collier d'épaule au lieu du cou a certes été conçu par l'homme mais une fois qu'il fut mis en place, des possibilités si nouvelles en ont résulté que plusieurs historiens y voient la cause de la disparition de l'esclavage et de l'éveil technique du Moyen Age. Tout au long de l'histoire on voit que la pensée imagine l'instrument, mais qu'en retour l'instrument conditionne la pensée.

Modèles et simulation

L'ordinateur a donc beau n'être qu'un exécutant, sa capacité, sa célérité, son zèle sont tels qu'il apporte au progrès des sciences une contribution décisive. Et pas seulement de la manière que nous avons vue jusqu'ici, c'est-à-dire en jouant complètement les partitions écrites par le savant mais trop lourdes pour son harmonica. L'ordinateur peut encore aider le savant à *écrire des partitions nouvelles*, et voici comment.

Imaginez que vous étudiez la circulation du sang dans les veines et les artères. Le sang est un liquide bizarre et compliqué, et qui change de teneur au cours de son périple. Et les conduits qui le guident, faits de peaux ultra-minces, sont encore plus étranges, sans compter le cœur qui le pousse à gauche et l'aspire à droite. Tout ça est si complexe que, cette fois, vous ne pouvez même pas *écrire* les équations du mouvement sanguin! Pour y voir quand même un peu clair, et pouvoir faire au moins quelques grandes prédictions (par exemple ce qui se passe si des graisses se déposent le long des parois, ou si telle admission à un rein se ferme légèrement) vous essayez de *réduire le phénomène à un autre plus facile à analyser*. Par exemple, vous assimilez tout le système circulatoire à une plomberie dont les tuyaux peuvent être décrits par, mettons, trois paramètres tels que diamètre, rugosité, élasticité, parcourue par un fluide caractérisable par, mettons, quatre paramètres tels que densité, viscosité, température, taux de particules en émulsion, et animée par une pompe aspirante-refoulante dont le mode d'action se résume par, mettons, trois paramètres tels que la période du cycle et les pressions au départ et au retour. Vous avez là élaboré ce qu'on appelle un « ***modèle*** » de votre phénomène, en ce sens que vous admettez que le sang circule dans son circuit réel à peu près ***comme*** le fluide dans votre circuit imaginaire. Et à quoi vous avance cette transposition? A ceci qu'alors que vous étiez étourdi et quasi paralysé par les mystères de ce vrai sang, poussé par un vrai cœur dans de vraies veines, vous êtes par contre en pays de connaissance avec votre fluide représentatif, poussé par une pompe dans une tuyauterie; car, de ce système-là, *vous connaissez*

les lois. Vous savez donc écrire les équations qui régissent le mouvement du fluide dans votre *modèle*, et quand vous les aurez résolues vous verrez si les propriétés trouvées, par exemple vitesse et pression aux divers endroits, concordent ou non avec celles que vous avez observées dans la circulation sanguine. Si oui, votre *modèle* est bon, du moins pour les propriétés qu'il vous a expliquées. Sinon, vous le retouchez en introduisant d'autres tuyaux, vannes ou paramètres, jusqu'à ce qu'il « *rende compte* », comme on dit, de tous les faits connus. Vous avez alors élaboré un « équivalent » du système réel sur lequel vous pouvez calculer et prévoir.

La « modélisation » est un procédé fondamental de la pensée scientifique, et même de toute pensée interprétative. Quand vous cherchez à comprendre les réactions d'un homme à une situation, ce que vous faites en réalité, c'est d'imaginer à sa place un être fictif simplifié, quasiment un mécanisme, réduit à un certain nombre de ressorts dont vous connaissez (ou croyez connaître) le jeu, tels que l'orgueil, la nervosité, les intérêts. Vous considérez que vous avez « expliqué » la réaction en question si ces ressorts, combinés en doses idoines selon des normes admises, produisent l'effet que vous avez observé sur l'homme réel. En un sens on peut dire qu'on fait un modèle dès qu'on traduit un phénomène en langage mathématique, car on isole alors dans ce phénomène des grandeurs qu'on suppose suffisantes pour le caractériser et entre lesquelles on sait qu'existent certaines lois préalablement dégagées. Ainsi, quand on déclare que les mouvements des astres sont régis par leurs simples masses et distances mutuelles, tout comme les mouvements de billes d'acier, on réduit à très peu de paramètres abstraits une situation concrète infiniment complexe, où nos ancêtres ont cru longtemps que jouaient aussi l'éclat du Soleil, le bleu du ciel, le froid de la Terre, les caprices d'Apollon... Les formules de la gravitation universelle de Newton « modélisent » donc fortement la réalité de l'univers stellaire.

Mais l'élaboration d'un modèle capable d'expliquer un ensemble de phénomènes devient une tâche monumentale dès que l'ensemble est tant soit peu complexe, parce que pour vérifier si un modèle se comporte en accord avec les phénomènes observés, il faudrait faire autant d'expériences qu'il y aurait d'hypothèses et de structures à essayer, et ce serait excessivement difficile, long et coûteux. Ici l'ordinateur vient nous apporter un secours décisif. On peut en effet **simuler** une expérience sur les constituants du modèle en représentant ceux-ci par des *symboles*, et écrivant un *programme* qui fait exécuter par l'ordinateur, sur ces symboles, des opérations *équivalentes* à celles que la nature exécuterait sur le modèle au cours d'une expérience réelle. On *connaît* ces opérations, puisqu'on sait selon quelles lois marche le modèle. La *simulation* est donc une sorte d'expérimentation au presse-bouton, immobile et mathématique, où le changement effectif de paramètres et de montages dans le modèle physique est remplacé par un simple changement de données dans le modèle codé, c'est-à-dire en fin de compte dans un calcul programmé. En un sens le programme dans l'ordinateur est un *modèle* mathématique du *modèle* physique. Et en un instant l'imprimante de l'ordinateur, ou mieux l'écran d'une console de visualisation, vous informe de l'effet que le changement aurait eu dans le modèle physique. Du coup, vous pouvez retoucher votre modèle de très nombreuses manières, jusqu'à ce qu'il devienne valable, très vite, sans bouger de votre chambre, et à peu de frais. Remarquez que même avec le secours d'une simulation sur ordinateur vous ne pourrez pas vous abandonner à la méthode paresseuse qui consisterait à parcourir toutes les valeurs imaginables des paramètres et toutes les facettes concevables des montages. Car, souvenez-vous-en, le nombre des

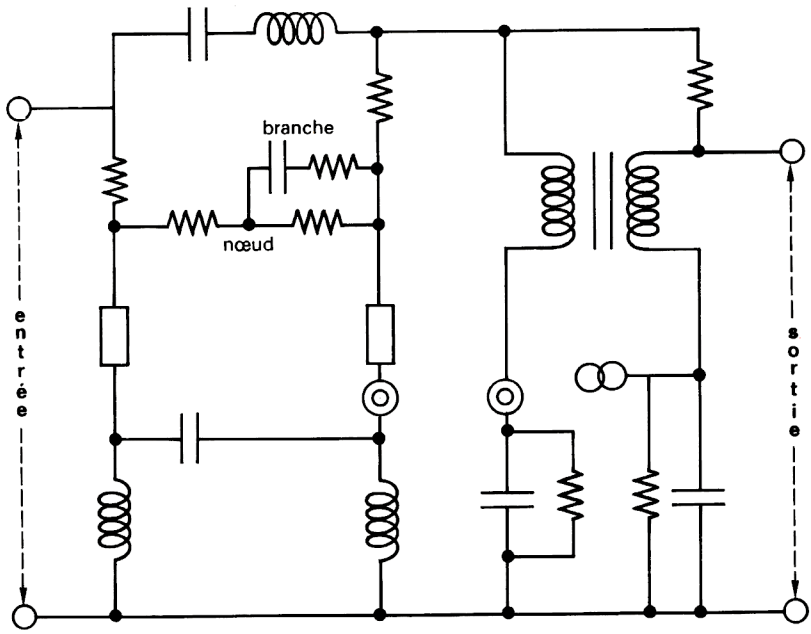
combinaisons possibles croît si fantastiquement vite qu'il vous faudrait des milliards de millénaires pour les parcourir toutes « telle la brute », même avec les modèles les plus simples. Pour conduire ces tâtonnements vers un terme utile dans un temps acceptable, votre esprit continuera donc d'avoir besoin de toute son acuité et de toute sa science, et même plus qu'avant l'ordinateur puisque celui-ci offre à son choix infiniment plus de chemins.

Le plus étonnant et le plus émouvant des centaines de modèles élaborés dans tous les domaines depuis l'avènement de l'ordinateur est celui... du cosmos! Eh oui, rien de moins que le cosmos entier. Dans une première expérience, on a lâché — en modèle, c'est-à-dire, en fait, simplement sous forme de paramètres représentatifs entrés en mémoire — une série de petites billes représentant les astres, que le programme soumettait aux lois de l'attraction universelle de Newton. On simulait toutes les positions et vitesses initiales imaginables, et on demandait à l'ordinateur de suivre les configurations qui allaient en résulter dans la suite du temps. Une minute dans l'ordinateur représentait un million d'années réelles. Le résultat fascinant fut que dans tous les cas les billes se groupaient en spirales très semblables... aux galaxies de notre univers! Dans une seconde expérience, on a essayé de retracer l'histoire des astres eux-mêmes. A cet effet, on a lâché dans l'ordinateur — toujours par symboles — des nuages et des poussières. Les nuages étaient formés de mélanges d'hydrogène et d'hélium ionisés ou non. Les poussières contenaient des grains minéraux. On faisait là encore de très nombreuses hypothèses variées sur la composition, l'ionisation et le mouvement de ce magma flottant dans son état initial. A partir de là, l'ordinateur ordonnait les positions et compositions futures selon les lois connues de la mécanique, de la chimie et de l'électricité. Et on suivait l'évolution du magma comme celle d'une émulsion agitée par une batteuse, toujours avec la minute valant un million d'années. Résultat encore plus extraordinaire, plus poignant que tout à l'heure : les états successifs obtenus, contés par l'ordinateur sur une imprimante d'abord, puis sur un écran cathodique, aboutissaient, au bout de cent milliards d'années... à la séparation du magma en groupes d'étoiles mères rayonnantes entourées de satellites qui ressemblent en tous points à notre système solaire! Ainsi l'ordinateur nous a donné, non pas certes la puissance proprement divine de *créer* à notre fantaisie un magma de taille cosmique, mais celle, plus humaine et tout aussi bouleversante, de *comprendre* comment s'est constitué le ciel étoilé qui brille au-dessus de nos têtes.

La conception assistée

Un autre service, voisin de la simulation et que l'ordinateur rend cette fois à la technique plutôt qu'à la science, est ce qu'on appelle la « *conception assistée* », sous-entendu conception de *systèmes* assistée par l'*ordinateur*. C'est là le royaume de la console de visualisation.

Prenons le cas d'un ingénieur ou savant qui essaie de mettre au point un circuit électronique complexe, où des composants variés sont insérés dans de nombreuses branches se joignant en de multiples nœuds (vous n'avez pas besoin de savoir ici ce que sont au juste les composants et ce que signifient les graphismes qui les représentent dans la figure ci-après). Notre homme connaît bien les équations qui régissent les courants dans les branches et les tensions aux nœuds, et qui permettraient de calculer



par exemple le signal qui sort du circuit en fonction du signal appliqué à l'entrée pour n'importe quelles valeurs des composants. Mais une fois de plus les calculs correspondants seraient infaisables à la main. Et s'en remettre à l'expérience ne vaudrait guère mieux, car si l'ingénieur voulait trouver empiriquement les valeurs optimales qu'il doit donner aux composants pour obtenir tel effet qu'il désire, il devrait faire un trop grand nombre d'essais. D'où résulte qu'en pratique, jadis, il pataugeait, tâtonnait au mieux de son intuition, faisait quand même quelques mesures, évitait de compliquer le circuit, et finissait par vendre une configuration modeste qui n'était fouillée qu'au dixième de ses possibilités et que le prochain concurrent surclassait.

Aujourd'hui, le même ingénieur s'assied devant une console connectée à la sortie d'un ordinateur. Il commence par informer l'ordinateur, via son clavier, de l'anatomie du circuit qu'il se propose d'étudier et de la manière dont il entend l'exciter, exprimant les nœuds, les branches et la nature des composants et des signaux d'entrée par des symboles standardisés. L'ordinateur contient un programme qui dresse d'une part le système d'équations régissant les courants et tensions dans cette anatomie, d'autre part un algorithme universel pour les résoudre, enfin des ordres graphiques à destination de la console et qui font instantanément apparaître le circuit sur l'écran avec des figurations convenues. Maintenant l'ingénieur donne par le clavier des valeurs aux composants, et touche avec un photostyl un nœud : instantanément lui apparaît sur l'écran, au-dessus de ce nœud, la valeur de la tension qui va y régner. Il change une valeur d'un composant, simplement au clavier, et aussitôt la nouvelle tension s'affiche. Toujours assis, sans trifouiller des montages, mais faisant voltiger ses touches et son photostyl, il passe d'un nœud à l'autre, explore toutes les branches, obtient même des renseignements plus profonds comme la sensibilité du signal sortant à l'égard de telle modification. Il a quand même besoin de toute sa science pour se

guider, car s'il s'avisait de parcourir sans aucune ligne directrice toutes les combinaisons possibles, leur fantastique multiplicité, que je vous ai signalée, le tiendrait sur sa chaise pendant quelques millénaires, en dépit de la supervitesse de l'ordinateur. Mais s'il est bon ingénieur, il aura dégagé le circuit vraiment optimum en quelques heures de domino là où jadis il lui fallait six mois au milieu d'une brousse hérissée de voltmètres et de fers à souder. Quel progrès en qualité, rapidité, économie!

Cette méthode de « conception assistée » s'étend rapidement à de nombreux domaines. Voici un architecte-urbaniste qui prépare une « cité ». Des analystes ont préalablement élaboré un vaste programme qui prévoit toutes les questions qu'on peut se poser à propos d'un avant-projet tracé dans ses grandes lignes : — ensoleillement des pièces, qu'on peut calculer en résolvant les équations des rayons à travers les cadres des fenêtres; — insonorisation en fonction de diverses compositions des murs mettant en regard les prix; — charges des ascenseurs pour diverses hypothèses sur la répartition des flots humains dans le temps et l'espace; — nombre des coudes de tuyauteries qu'implique tel ou tel déplacement des cuisines et salles d'eau; — dépendance du prix de revient à l'égard de telles variantes; etc. Ce programme étant réalisé, et mis en machine, l'architecte s'assied devant la console, et commence par se faire afficher l'esquisse, en simili relief, de telle ou telle partie de son ensemble. Puis il touche du photostyl une fenêtre, l'ordinateur calcule en un clin d'œil l'éclairement à travers elle, et les zones de lumière et d'ombre apparaissent dans la chambre située derrière la fenêtre. Il essaie une amélioration en proposant un élargissement ou un déplacement du cadre, ou un écart différent de la maison d'en face, et chaque fois l'effet se dessine sur l'écran à peine la question posée. Il peut même demander à voir une porte, ou un globe, à partir de points différents, et l'ordinateur résoud en une seconde les équations des perspectives correspondantes et présente les divers aspects à l'architecte. C'est un véritable régal pour l'œil et l'esprit que d'assister à cette danse des formes avec la pensée, où la pensée agile, pertinente, inquiète, appelle et entraîne comme par une sympathie amoureuse des formes merveilleusement précises, fines, sûres et pures. N'allez pas croire que ce n'est là qu'un affichage de ce qu'il avait fallu prévoir dans le programme. Celui-ci ne contient que les *algorithmes* pour calculer les réponses aux questions posées, et ces algorithmes doivent tourner à neuf pour chaque question et chaque donnée nouvelles introduites par le meneur du jeu. Ce sont donc bien des résultats originaux qu'on visualise à chaque coup, seul le processus pour les obtenir est le même et a été préparé à l'avance.

C'est ainsi encore qu'avec une console au bout d'un ordinateur convenablement programmé, on met au point, dans les secteurs de pointe de l'industrie :

— les plans de câblages, d'imprimerie, de masques pour les circuits électroniques plus ou moins intégrés;
— les profils des ailes d'avions devant des flux d'air simulés;
— la visibilité à partir de postes de pilotage diversement meublés;
— les formes optimales des carrosseries d'automobiles;
— les profils des cames en fonction des déplacements recherchés;
— les systèmes optiques où le parcours des rayons est visualisé;
— les récipients de plasmas nucléaires;
— le planning d'une production complexe.

Remarquez qu'il ne faut pas non plus que l'admiration pour ces méthodes nous fasse oublier le précepte de Montaigne qu'en toutes choses, fût-ce en face des merveilles, il faut raison garder. Le couple ordinateur-console n'est qu'un outil, même lorsque ses services ressemblent aux prodiges que les héros des Mille et Une Nuits obtenaient des génies esclaves qu'ils délivraient d'une bouteille. Il aide puissamment le concepteur à optimiser et décider, mais reste en fin de compte, à travers le programme, tributaire de bases scientifiques que des cerveaux humains ont dû assembler au prix d'efforts parfois séculaires. Au départ de toute nouvelle entreprise, les lois, les données, les matériaux, les procédés, doivent toujours être conçus, essayés et codifiés par des spéculations d'hommes et des expériences en vraie grandeur. Mais une fois ces fondations bien assises, l'ordinateur permet d'édifier dessus bien plus de palais qu'il ne semblait possible avant lui.

Les sciences nouvelles de l'homme

Dans certains domaines, les calculs nécessaires pour mettre en nombres des principes connus sont trop lourds pour l'harmonica humain, non pas parce que les algorithmes qui les commandent sont trop complexes, mais parce que *les données à malaxer sont trop nombreuses*. C'est le cas notamment de la *statistique*.

Les comportements des individus et de leurs groupes sociaux dépendent de milliers de petites influences n'obéissant qu'au hasard, ou à des causalités et motivations si multiples, subtiles et enchevêtrées que c'est tout comme. De ce fait, les quelques régularités qu'on peut y déceler n'ont plus l'aspect des *lois* de la physique, mais sont de types « probabiliste » et « statistique ». Ainsi les habitudes du public en matière de magazines semblent totalement aléatoires quand on les enregistre sur les vingt voisins du métro. Mais si on en étudie des centaines de mille et qu'on les compare aux statuts socio-professionnels des interviewés, on finit tout de même par constater, par exemple, que contrairement aux idées reçues, les jeunes n'achètent pas plus de magazines sportifs que les vieux, et qu'en accord avec les idées reçues, les intellectuels achètent moins de magazines illustrés que les commerçants. Remarquez que la nature ne nous présente des lois « exactes » formulables en équations mathématiques que lorsqu'on l'étudie à l'échelle macroscopique, à laquelle un caillou apparaît comme un objet unique et d'un seul tenant. Mais si on veut analyser le caillou microscopiquement, comme un assemblage d'atomes, le désordre de fond ressurgit, et les comportements des particules infinitésimales ne peuvent à nouveau être appréhendés que par des moyennes statistiques.

La statistique, contrairement à un préjugé courant, est l'instrument fondamental, irremplaçable, quasiment unique, de l'étude des phénomènes humains, sociaux et microscopiques. Depuis le début de ce siècle (et cela continue) une pléiade de mathématiciens ont centuplé sa puissance. Ils ont mis au point des méthodes et algorithmes qui permettent, en principe, de découvrir, dans des masses de caractères aléatoires relevés sur des populations énormes, s'il y a ou non des « *corrélations* » entre les caractères, ou des « *facteurs* » communs, ou des « *formes* » dominantes. Pour ne citer qu'un exemple, si on recueille les scores obtenus par dix-mille enfants d'un certain groupe sur une batterie de trente tests mentaux, on peut par ces méthodes, en principe, déceler si toutes les variations observées sont « explicables » par le jeu d'un petit

nombre de causes. On en a trouvé une en particulier qu'on appelle « le facteur *g* », qui rend compte à lui seul de plus de 50 % de ces variations. C'est donc la preuve qu'il existe dans tous les esprits quelque chose qu'on peut appeler « l'intelligence générale », — et c'est la première lettre de l'adjectif « générale » qui a donné son nom à cette source commune de succès ou d'échecs dans les tests. Il reste aux psychologues de l'esprit à scruter et interpréter le facteur *g*. De même, en étudiant par ces méthodes (qu'on appelle « analyse des variances multiples ») les rendements de milliers de sols pour des dizaines de plantes, on découvre qu'il existe un facteur commun qui les influence tous de façon appréciable, et qu'on doit considérer comme une vertu générale de « productivité agraire », vecteur d'une nouvelle discipline.

Mais ici comme dans les autres sciences, c'est une chose d'avoir inventé les algorithmes savants qui permettent en principe de dégager quelque structure dans les tourbillons des mesures statistiques, et c'en est une autre de les exécuter jusqu'au bout, à cause des torrents de données à manipuler. Il y a donc encore moins d'espoir (si possible) de jamais les terminer à la main. D'où à nouveau l'intervention de l'ordinateur dans le même rôle d'exécutant hyperrapide. Cette fois la terminaison d'un programme de dépouillement dure non plus des heures, mais des jours. Les gros ordinateurs employés pour moudre les grandes statistiques humaines, économiques et sociales tournent pratiquement sans arrêt toute l'année. On en sort tout de même des résultats de plus en plus intéressants, qui commencent enfin à jeter quelque lumière sur cette composante la plus mystérieuse du progrès : son auteur. Et aussi sur ses œuvres les plus subtiles, telles que le langage, qui semblaient échapper à toute analyse systématique. Or, avec l'ordinateur on peut conduire jusqu'au bout toutes sortes de corrélations et analyses statistiques sur les fréquences d'emploi des mots, les fréquences de leurs couplages à deux ou à trois, la nature de leur « environnement », les cas d'inversions, interrogations, appositions, invocations, les points où une phrase se ramifie, etc. Et à partir de là, on a commencé d'apercevoir quelques pâles lueurs dans la formidable forêt du verbe. A travers le monde, une trentaine de gros ordinateurs travaillent sur ces thèmes 365 jours par an depuis 5 ans, et on peut raisonnablement espérer, après encore une vingtaine d'années de ces efforts, que l'homme commencera à savoir un peu comment il parle, et qu'une linguistique sera née. Vous avez vu qu'il en est de même dans les sciences médicales et dans l'enseignement.

En économie, l'ordinateur sert non seulement à exploiter les statistiques, mais encore à résoudre certains de ces problèmes de choix ou de décision qui défient les intuitions des meilleurs dirigeants. Où faire arriver les méthaniers, au Havre qui est plus loin des sources mais qui possède une raffinerie, ou à Marseille plus proche mais d'où il faudra envoyer les chargements se faire raffiner ailleurs ? A quel projet d'avion faut-il accorder des crédits : Concorde, Mercure ou Airbus ? Comment planifier une activité à multiples branchements comme la construction d'un sous-marin nucléaire, et même, dans un cadre plus étroit, comment faut-il répartir les charges de travail des divers postes d'une usine qui ont des performances variées de façon que le prix de revient soit minimum, et que faire si ce prix entre en conflit avec par exemple le délai de fabrication ? Les questions de ce genre, qui forment le pain quotidien de l'économie, sont terriblement difficiles non seulement par leur texture même, mais encore parce que les paramètres qui interviennent dépendent de très nombreuses incidences extérieures, et, pire encore, de l'avenir incertain. Car telle performance d'un avion qui vaut 5 sur 10 aujourd'hui, vaudra peut-être 2, ou 8, dans quelques

années si l'insonorisation des appartements s'est améliorée, ou que des aérotrains lient les aéroports aux centres des villes, ou que le tourisme en bateau ait repris de l'attrait. En outre, les critères mêmes des décisions entre lesquelles on balance sont rarement clairs et sûrs, les conditions que le décideur s'impose étant souvent des *inégalités* plutôt que des *équations* (par exemple, je veux que telle machine travaille *moins de* 5 heures par jour).

En dépit de toutes ces occasions de migraines, les chercheurs modernes ont courageusement mis ces problèmes en chantier. Ils ont élaboré divers corps de doctrine, qui s'appellent la « *recherche opérationnelle* », la « *théorie des jeux* », « *l'analyse des unités économiques par les matrices d'entrées-sorties* » (matrices de Leontieff) et qui permettent quand même d'écrire, pour les principales variables d'un problème économique, des équations assorties de tout un cortège de « liaisons », d'informations statistiques et d'hypothèses probabilistes. Dans certains problèmes on est obligé d'écrire des milliers d'équations simultanées pour des milliers d'inconnues avec des paramètres aléatoires. Ç'aurait été folie pure, ambition dérisoire voici 15 ans. Depuis l'ordinateur, il n'y a plus de raison de se priver. Il finira par résoudre le système, fût-ce en tournant un mois... ça ne sera toujours pas cent mille ans.

On peut dire, sans aucune enflure rhétorique, que l'ordinateur permet enfin aux sciences de l'homme et de la société de démarrer sérieusement. Il aura été dans ce domaine ce que la lunette de Galilée fut à l'astronomie et le charbon à l'industrie. La plus grande peut-être des promesses qu'enferment ses électrons harnachés, c'est que va cesser enfin cette situation affligeante où l'homme sait tout sur ses instruments et rien sur son âme, ce qui fait ressembler son appareil de production à un immense attirail de leviers actionnés par des spectres. Mais c'est peut-être aussi la cause de la sourde inimitié de tant de gens contre l'ordinateur. L'homme, ancestralement, a eu peur de se connaître lui-même. Espérons que l'ordinateur l'aidera à vaincre peu à peu cet ultime obstacle à la paix qu'il faudra bien qu'il établisse avec son être comme avec la société et avec l'univers.

L'on a vu que, grâce à l'ordinateur, la science peut calculer jusqu'au bout les effets de bien des phénomènes ou arrangements techniques dont elle connaissait les lois mais qu'il était impossible d'expliciter numériquement. Moyennant une courte incursion dans le concept d'équation, vous avez appris que la science peut en écrire bien plus qu'elle n'en sait résoudre *par des formules* mais que l'ordinateur peut les résoudre *point par point* grâce à des *approximations successives,* en des temps acceptables et qui sans lui seraient prohibitifs. Puis on vous a invité à entrer dans l'univers sans cesse plus peuplé des *modèles* et des *simulations,* assistés par l'ordinateur et dont savants et techniciens se servent pour étudier les systèmes réels sans avoir à les monter et mesurer effectivement. Ce qui vous a conduits à voir des ingénieurs faire le projet d'installations considérables en quelques heures, assis devant une console de visualisation servie par un ordinateur. Pour finir, on vous a déclaré que l'ordinateur est le seul instrument capable de faire enfin démarrer une vraie science de l'homme, de l'homme seul ou de l'homme en groupe, car cette science ne peut se fonder que sur des statistiques très abondantes, et l'ordinateur est seul à pouvoir brasser des grandes masses de chiffres avec la vitesse requise.

14

L'économie politique de l'ordinateur

L'industrie et son chiffre d'affaires

Nous avons parcouru au cours du chapitre précédent une série d'applications de l'ordinateur dont vous avez constaté que toutes étaient bénéfiques, pour l'horizon intellectuel des hommes autant que pour leur prospérité. La série ne couvre qu'une très petite portion de ces applications. En fait, on trouve aujourd'hui l'ordinateur en bonne place dans la quasi-totalité des activités humaines dès qu'elles dépassent le niveau artisanal, et encore beaucoup d'artisans se groupent-ils en coopératives qui tiennent sur ordinateur des listes de fournisseurs, de prix, de taxes, etc. Le présent livre, et ceux qui vont encore être écrits dans les deux ou trois prochaines années, seront peut-être les derniers qui détailleront des applications une à une. L'ordinateur entre autant dans les mœurs que l'électricité ou le tour mécanique. Personne ne songe plus à égrener des listes de cas où on emploie ces deux moyens d'action; chaque industriel en a chez lui pour faire ce qui lui convient. C'est un outil à vocation universelle dont bientôt on apprendra à se servir dès l'école.

Il y a aujourd'hui à peu près 200 000 ordinateurs de toutes tailles installés de par le monde, dont 100 000 aux États-Unis, 15 000 au Japon, 15 000 en URSS et pays de l'Est, 70 000 en Europe Occidentale (18 000 en Allemagne, 14 000 en Grande-Bretagne, 12 000 en France, 8 000 en Italie, 6 000 au Benelux, etc...)

L'industrie américaine investit en ordinateurs 10 % de tout ce qu'elle investit en biens d'équipement.

Le chiffre d'affaires total de l'industrie qui fabrique le hardware est de l'ordre de 45 milliards de dollars, et s'approche de celui de l'automobile. Ce qui est phénoménal si on songe que l'industrie n'a que 20 ans et ne vend pas aux particuliers. Le taux de croissance du parc ordinateurs et du chiffre d'affaires correspondant est de 15 à 20 % par an depuis 10 ans, taux qui a été rarement atteint dans quelques secteurs explosifs. Et nous avons vu que l'informatique conserve la palme de l'innovation. C'est dans ses sociétés qu'on trouve le plus grand rapport, de très loin, entre travailleurs de matière grise et travailleurs manuels.

Il devient d'ailleurs de plus en plus difficile de parler de l'ordinateur comme d'une unité en soi. C'est un hall de machines qu'on installe, une « configuration » comme on dit dans le métier, plutôt qu'une machine. On y trouve un très grand nombre de sous-unités diverses, souvent fabriquées par des usines indépendantes et vendues séparément : les consoles, les meubles de rangement, les câblages, les équipements pour la transmission de données à distance, et les multiples périphériques et

interfaces qui font l'objet de plus en plus d'une industrie séparée de celle qui fabrique l'unité centrale de traitement. L'informatique suit ici l'évolution de toutes les grandes industries, elle s'étale, se diversifie, se banalise, et s'insère dans le « magma » général par d'innombrables fils. Elle offre aussi une gamme de plus en plus variée de modèles, depuis le tout petit ordinateur nichant dans une machine à calculer de bureau, jusqu'au super-géant 6 600 de la Control Data Corporation qui occupe un petit immeuble, abrite des millions de mots en mémoire statique et des milliards en séquentiel, dont le temps machine est de 0,2 microseconde, et le tout à l'avenant. Une partie croissante de l'industrie fournit des pièces de rechange et des unités de remplacement, car les systèmes de base se font de plus en plus « modulaires » afin qu'on puisse les bâtir, les étendre ou les renouveler brique à brique.

La gamme des prix s'étend de 200 000 F pour les plus petits calculateurs de poche jusqu'à 200 000 000 de F pour le 6 600 de la CDC. Mais beaucoup d'usagers *louent* leurs calculateurs au fabricant.

L'industrie de l'informatique engendre aussi des retombées importantes dans toutes les autres industries, depuis la chimie jusqu'au cinéma, en passant par la mécanique et, naturellement, l'électronique, dont elle absorbe aujourd'hui 70 % des travaux du secteur dit « professionnel ». C'est, sans conteste, la plus dynamique et déjà parmi les plus grosses des industries modernes.

Elle présente encore un caractère étonnant : une seule société IBM, y détient 65 % du marché. La plus grosse après elle, UNIVAC, en détient 8 %. On voit que la prépondérance d'IBM est unique en son genre. Le grand public appelle souvent « IBM » tout équipement d'informatique, même s'il est en fait d'une autre marque comme le nom « gramophone », originairement celui d'une société, s'était confondu avec la notion même d'un reproducteur de son.

La saisie des données

L'un des domaines inattendus que l'explosion de l'informatique aura avivés, voire engendrés, c'est celui qu'on pourrait appeler... de la poinçonneuse. Il ne faut pas oublier, en effet, que toute l'informatique se base sur la possibilité de réduire les problèmes à des traitements programmables de *données*. Encore faut-il que les données soient saisissables. Et c'est là, tout à fait au début de la chaîne d'informatique, et même plus exactement en amont d'elle, qu'apparaît, sous une forme naine et fantasque, l'une des difficultés majeures, souvent énervante, de ce nouvel art.

On pourrait rendre automatique et immédiat le calcul de la TVA maudite si les vente des épiciers n'étaient pas inscrites au crayon sur de méchants bouts de papier que le vent emporte aussitôt le paiement effectué. On pourrait régler tellement mieux la circulation si les voitures étaient proprement comptées aux entrées et sorties des carrefours. On pourrait apprendre des vérités essentielles sur l'homme et la guerre si au cours des manœuvres militaires les gestes des soldats, les visions des observateurs, les cris des officiers, étaient enregistrables en code à leurs sources. Et ainsi de suite. Le goulot d'étranglement de l'informatique n'est plus le hardware, ni le software, c'est la *saisie des données*. Jadis on pouvait dire que l'ordinateur s'évaluait par la qualité de ses circuits ou des programmes. Aujourd'hui circuits et programmes sont

bons, et il faut dire : « tant valent les données, tant vaut l'ordinateur ». Le plus intelligent, le plus efficace programme du monde dans la meilleure des machines ne peut rien si les données sont pleines de trous et d'erreurs. La plupart des ratés qu'on observe encore en informatique vient de ce que les données ne sont pas recueillies assez complètement, proprement et rapidement.

Pour s'attaquer à ce problème, on doit faire appel à des moyens qui se situent à l'extrême opposé de la haute science employée en hardware et software. Il faut d'abord savoir discipliner, sans empoisonner la vie des gens, toute une série d'actes qu'ils accomplissent quasi d'instinct, et pour cela déployer toute la stratégie patiente du bon père de famille qui veut obtenir de son petit garçon qu'il ne traverse qu'entre les clous. Il faut y mélanger le génie de l'organisation et l'aptitude à éduquer et persuader. D'autre part, et nous parvenons à la poinçonneuse, il faut inventer et mettre en œuvre d'innombrables petits engins, infiniment astucieux mais infiniment simples à manier, capables de transformer aussitôt que possible les actes humains générateurs de données en signes ou objets interprétables par les machines d'informatique. Leur prototype est la poinçonneuse qui traduit chaque personne ayant traversé un portillon par un trou. Les portillons automatiques ont une poinçonneuse incorporée animée par le petit tourniquet que vous poussez. La gamme de ces gadgets est inépuisable et donne lieu à un artisanat nouveau et florissant. Cela va des formulaires automarqués, en passant par les plus amusantes formes de crayon, griffes et timbres, aux anneaux qui inscrivent le code de celui qui les porte au doigt quand il l'applique sur une case préparée d'un carton. Voilà qui nous ramène au Moyen Age où des moyens de ce genre servaient à identifier des hommes ne sachant pas écrire. Curieuse jonction de l'extrême de la sophistication scientifique avec les petites astuces de nos ancêtres. Les innombrables petits capteurs qui saisissent les données dans leur berceau sont à l'ordinateur ce que sont à l'arbre les racines et radicelles qui vont chercher les sucs de leur vie dans la terre.

Les métiers de l'informatique

L'entrée de l'informatique dans tous les domaines d'activité a naturellement fait surgir plusieurs métiers nouveaux.

En gros, on peut distinguer cinq niveaux dans la hiérarchie des fonctions des gens qui s'occupent d'un département « informatique » dans une société, banque, hôpital, usine, armée, port, centre documentaire et autres.

Au premier niveau on trouve des *maîtres d'œuvre*, appelés aussi « concepteurs de systèmes », qui doivent concevoir de haut les tâches à faire exécuter par l'ordinateur, l'insertion économique et technique de ce dernier dans l'ensemble des activités, ses incidences et son développement. Ce sont presque toujours des ingénieurs des grandes écoles ou licenciés ès sciences.

Au-dessous, les *analystes* reçoivent du « maître d'œuvre » des problèmes déjà circonscrits, mais où les objectifs et le cadre sont encore formulés dans le vocabulaire « naturel » de l'entreprise. L'analyste doit transposer ce problème en termes d'informatique, mais seulement dans les grandes lignes. Il trace en somme l'organigramme des opérations qu'aura à accomplir l'ordinateur, établit le codage, vérifie s'il disposera

des données nécessaires (sinon fait prendre les mesures adéquates par le maître d'œuvre), estime le temps et le coût des calculs, définit la forme sous laquelle les résultats devront être livrés pour obtenir une exploitation optimum. Si le problème posé implique des équations mathématiques complexes, c'est l'analyste encore qui doit préparer les algorithmes pour les résoudre. Mais on rencontre rarement ce cas dans les problèmes de gestion, de sorte que les analystes de ce vaste domaine n'ont pas besoin d'une formation mathématique poussée. L'analyste doit être surtout un « explorateur logicien ».

L'analyste passe le travail, ainsi défini dans ses grandes lignes et bien ordonnancé, au *programmeur*, à qui incombe la tâche (ingrate!) de réaliser le programme effectif, cette fameuse liste parfaitement précise et rigoureuse, sans le moindre petit trou ni la plus minime bavure, de toutes les instructions commandant tous les pas du traitement, lesquels peuvent se compter par milliers. Vous vous rappelez qu'il dispose à cet effet de « langages » de programmation évolués. Même avec ce puissant secours, sa tâche exige une patience, un scrupule, une minutie extrêmes. Et même quand il dépense des trésors de ces qualités, il se trompe souvent, et une partie importante de sa responsabilité est de *vérifier* le programme, en le faisant tourner sur l'ordinateur, pour détecter et corriger tous les passages fautifs. On compte en général *un* temps pour la première écriture d'un programme et *trois* temps pour les essais jusqu'au programme validé. Sauf pour les applications scientifiques, le programmeur n'a pas du tout besoin de connaître les mathématiques, il doit par contre connaître à fond la science de la programmation, le software de sa machine et le marché des sous-programmes préfabriqués disponibles. Le métier convient bien aux femmes. Un baccalauréat ou un CAP suffisent pour devenir programmeur après un an d'études.

Une fois le programme terminé et validé, les *exploitants* s'en emparent, le font entrer dans l'ordinateur, récoltent les résultats, surveillent et réparent les machines. Ce sont donc des techniciens du hardware, analogues aux conducteurs de pelles sur les chantiers, en général du niveau agent de maîtrise ou sous-ingénieur électronicien.

En plus, un département d'informatique emploie de nombreux *auxiliaires* du niveau secrétaire-dactylo pour perforer les cartes, taper des états, tenir les bibliothèques, etc.

La demande de l'économie pour ces personnels ne cesse de croître. Le « Plan Calcul » français a estimé comme suit les besoins pour 1975 : 40 000 maîtres d'œuvre, 40 000 analystes, 80 000 programmeurs, 60 000 exploitants, 160 000 auxiliaires, soit au total 380 000 spécialistes, plus que n'en emploie l'industrie textile pourtant vieille de 200 ans et qui habille 50 millions de gens. Le manque de ce personnel qualifié pèse lourdement sur le développement de l'informatique en France.

L'informatique n'est pas devenue seulement, en 20 ans, l'une des plus grandes industries modernes. Elle a fait une entrée aussi fulgurante dans le commerce de plus en plus important des « services intellectuels ». On compte aujourd'hui par centaines, dans chaque grand pays industriel, les sociétés indépendantes — parfois aussi des départements chez les constructeurs — qui proposent aux entreprises d'exécuter pour elles des études complètes leur permettant de « mettre en informatique » tout un secteur d'activité. L'étude se termine, si le client le désire, « clé en mains », c'est-à-dire par l'ordinateur installé et roulant chez lui, avec tout son personnel formé. Dans la même branche des « services » il faut compter aussi la confection des programmes préfabriqués, dans diverses langues, qu'on vend sur bande ou disque à des

prix allant de 1 000 F à 500 000 F, selon la taille et la complexité. On édite aujourd'hui plusieurs catalogues des programmes et sous-programmes disponibles, qui contiennent des centaines de titres. Enfin, les plus grosses de ces sociétés de service offrent de faire faire le traitement dont un client peut avoir besoin sur leurs propres machines. Avant que nous examinions d'un peu plus près ces centres, sachez que l'ensemble des activités de service tournant autour de l'informatique, atteint un chiffre d'affaires, pour l'ensemble du monde, de l'ordre de 30 milliards de dollars, soit les deux tiers du chiffre d'affaires de la production proprement dite du hardware. C'est une situation analogue à celle de l'industrie automobile, qui a induit des circuits d'affaires annexes (garages, pièces détachées, stations-service, tourisme) aussi importants économiquement qu'elle-même.

La déesse aux vingt bras

Alors que l'introduction de l'informatique dans les entreprises sera de plus en plus faite par leur propre personnel, il restera toujours, pour raisons de rentabilité, une grande place pour les centres qui font de l'informatique à façon, en coup par coup, à l'usage de clients extérieurs, un peu comme les scribes publics de jadis écrivaient pour le public analphabète les lettres convoyant les messages qu'on leur définissait oralement. Ces centres offrent une grande variété de services. Vous pouvez leur envoyer votre problème, formulé en langage courant, et ils se chargent de toute la suite des travaux d'analyse, de programmation, de traitement, de présentation des résultats. Ou bien vous pouvez leur envoyer un programme tout rédigé, dans un langage qu'ils acceptent, et ils mettent simplement à votre disposition l'ordinateur, dans une sorte de location sans chauffeur. Vous pouvez aussi demander des flashes de traitements partiels limités à telle ou telle phase d'une tâche complexe. Vous pouvez enfin demander tous ces services par téléphone, recevoir la réponse de même, et, si besoin est, alimenter aussi par fil l'ordinateur du centre avec des données fraîches, comme par exemple quand vous demandez à un centre de tenir votre stock en lui communiquant tous les soirs entre 21 h et 21 h 30 les états des achats et ventes de la journée. L'un des secteurs les plus prospères dans ce domaine est la comptabilité à façon pour les petites entreprises, sur des programmes universels tenant compte des règles de la comptabilité nationale que si peu de gens connaissent. Les tarifs sont élaborés dans les moindres détails : tant de centimes pour la seconde de passage sur l'unité centrale, tant pour la minute sur l'imprimante, tant le mot pour la transmission à distance, etc. C'est pire que le barème des carrossiers qui distingue 250 types d'intervention.

La prestation de ces services importants a posé des problèmes considérables sur tous les plans. D'abord en software, où il a fallu développer des programmes aussi polyvalents que possible afin de servir au moindre prix une multitude de clients posant des problèmes analogues avec seulement des données ou des contraintes différentes. Ensuite il a fallu fabriquer des « *terminaux* » pour permettre aux clients lointains de communiquer avec un centre, dans les deux sens, c'est-à-dire d'informer l'ordinateur et de se faire informer par lui. On trouve dès aujourd'hui sur le marché, en location ou en vente, une vaste gamme de terminaux, allant de la machine à écrire

électrique jusqu'à la console de visualisation en passant par les téléscripteurs de toutes complexités. Car les travaux de conception assistée que je vous ai décrits plus haut, vous pouvez les faire désormais assis dans votre studio avec l'ordinateur à cent kilomètres de là.

Enfin, pour que les Centres de traitement à façon puissent travailler de manière commode et rentable, il fallut les mettre à même de *servir plusieurs clients à la fois*, et ce besoin nouveau fut la source d'importants perfectionnements par lesquels nous allons clore notre voyage.

Dans un premier temps, on s'est contenté de déposer les problèmes et leurs programmes (sur bandes) dans ce qu'on pourrait appeler des « paniers » séparés, et de créer un superprogramme « de régie ». Ce superprogramme instruit la machine de parcourir les programmes en attente, de prendre note des traitements qu'ils requièrent, et de les découper en tranches qu'il appellera dans un ordre tel que l'occupation des organes de l'ordinateur soit optimale. Par exemple, une tranche d'un problème qui requiert une passe d'impression lente est transférée sur une mémoire séquentielle ordinaire pour être exécutée en même temps qu'une tranche d'un autre problème tourne sur la mémoire statique centrale. On peut tenir compte, en plus du critère d'occupation optimale des organes de traitement, d'indices accordant à tel problème une priorité sur tel autre. Cette méthode s'appelle en américain « *batch processing* ». Elle n'exige pas des ordinateurs spéciaux.

En franchissant un gradin de plus, on arrive à ce qu'on appelle la **multi-programmation.** Dans ce cas on introduit simultanément en mémoire tous les programmes des clients. A un moment donné, un seul de ces programmes tourne effectivement, c'est-à-dire anime les unités de traitement ou les interfaces, mais on peut l'interrompre au profit d'un autre qui passera une tranche, puis revenir au premier si c'est opportun, ou sauter à un troisième. Comme dans le cas précédent, il faut un superprogramme de régie, appelé ici « **moniteur** », pour décider ces tours de piste, mais le déroulement est plus rapide puisqu'on ne sort plus de la machine pour aller cueillir sa nourriture dans tel ou tel panier. Mais cela se paye par une complexité plus grande du moniteur. Il faut en effet assurer de nombreuses fonctions de police pour éviter que les tranches appelées ne se chevauchent, par exemple, protéger la zone de mémoire dans laquelle siège l'un des programmes contre l'envahissement par des résultats transitoires de calculs engendrés par un autre programme, ou organiser un contrôle rigoureux des interruptions par rapport à l'horloge interne. Vous aurez une idée du raffinement que cela exige, tant en software qu'en hardware, si vous songez qu'une tranche d'un programme peut solliciter, mettons, la lecture-écriture d'une pile de disques. Or le positionnement d'un bras prend une demi-seconde, pendant laquelle l'horloge interne débite 500 000 tops, et dans cette cataracte il faut isoler les tops exacts qui marqueront la « fin » d'un retrait du bras ou le « début » d'une sortie, ou la fin et le début d'une lecture de bloc. Il faut aussi prendre soin que les programmes à effectuer remplissent la mémoire avec le maximum d'économie, et gérer au vol, et au mieux, les périphériques qui sont mis en pool, en particulier les imprimantes qui impriment des résultats partiels répondant à des problèmes divers. Et le plus difficile est encore de bien gérer les tâches à accomplir, non seulement au jour le jour mais encore à long terme. Le tout conçu et agencé de telle manière que le client ne soit jamais conscient du fait que d'autres programmes sont traités en même temps que le sien, ce qui est surtout nécessaire lorsqu'il veut disposer d'un mode conversationnel. Les moniteurs de la

multiprogrammation sont en fin de compte si complexes qu'il est impossible de dire s'ils sont bons ou mauvais rien qu'en les lisant. D'ailleurs l'évaluation de la qualité, dans les gros programmes, est l'un des problèmes encore non résolus de l'informatique.

A un gradin encore au-dessus de la multiprogrammation on trouve ce qu'on appelle le « **multitraitement** ». Il combine l'éclatement du software en plusieurs ruisseaux simultanés avec un éclatement du hardware. La « configuration » employée ne ressemble alors plus du tout à « un » ordinateur. Elle peut comprendre plusieurs unités de traitement, plusieurs blocs de mémoire statique, et de nombreux « guichets » où on vient déposer ou recueillir l'information allant vers les périphériques ou en venant. Ces « guichets » sont en fait à leur tour de vrais petits ordinateurs, qu'on appelle « *unités d'échange* » pour les distinguer des interfaces simples, ou encore « *canaux* ». On doit loger là des moyens de calcul, car les périphériques sont banalisés (c'est-à-dire mis à la disposition de quiconque les appelle) et il faut donc savoir quand on peut y déverser ou y puiser de l'information, en tenant un compte exact, à la milliseconde près, de tous les espaces et de tous les temps disponibles, ainsi que de tous les besoins d'entrées et de sorties que formulent les processeurs. La gestion de ces échanges exige des calculs importants et des programmes en conséquence. Certaines installations pour multitraitement sont capables de servir jusqu'à 30 postes, munis de consoles de visualisation, postes qui lui posent tous des questions à n'importe quel moment, et reçoivent les réponses dans les quelques secondes qui constituent le « tac au tac » à l'échelle humaine, en ayant l'impression qu'ils sont seuls à correspondre avec la machine.

Encore un degré au-dessus on parvient dans le rêve. Les extrémistes ont découvert en effet que, même en multitraitement, la plupart des circuits des processeurs restent oisifs, la plupart du temps. Ils voudraient saupoudrer les tâches de façon si fine, en pénétrant si profondément dans le ventre des machines, que chaque cellule ait quelque chose à faire à chaque instant, par exemple qu'un additionneur traite déjà le digit d'un programme pendant que l'accumulateur envoie dans la mémoire une somme formée au service d'un autre programme. L'appellation « temps partagé » (en anglais « time sharing ») fut forgée voici un lustre pour désigner ce type de fonctionnement. Comme il s'est révélé jusqu'ici irréalisable, le terme a peu à peu glissé pour couvrir les autres procédés que je vous ai décrits permettant de prendre en charge simultanément plusieurs tâches. Mais le rêve subsiste chez les plus ambitieux des informaticiens, qui entendent jouer à plein leur rôle de démiurges.

Les installations à multitraitement ressemblent à ces divinités hindoues nanties d'une vingtaine de bras qui s'étendent dans tous les sens pour saisir ce dont leur âme a faim. Dans le cas de l'informatique, les bras cherchent des problèmes, les problèmes de ces hommes qui ont conçu la divinité. Pour leur plus grande gloire ou pour leur plus sombre détresse? C'est ce qui nous reste à voir dans la conclusion.

On a brossé le panorama de l'informatique en tant qu'industrie, et l'on a appris qu'elle est devenue en 15 ans l'un des secteurs de base de l'économie.

Conclusion

Y a-t-il révolution ?

La vie va-t-elle se robotiser ?

Est-on menacé de chômage ?

Est-on menacé de tyrannie ?

L'ordinateur ravira-t-il à l'homme la couronne de l'intelligence ?

... Où on essaie de dissiper les nombreuses fantasmagories
qu'on cultive autour de ce fait considérable
mais simple : les hommes disposent
désormais d'instruments capables d'exécuter un
million de fois plus vite qu'eux-mêmes
les traitements d'informations qu'ils leur ont préparés.
D'où résulte que l'ordinateur
ne peut que faire progresser la civilisation ;
et, contrairement à ce qu'on pense souvent,
dans un sens humaniste.

Y a-t-il révolution?

L'idée se répand et s'impose partout que l'informatique est une révolution. Est-ce vrai?

L'informatique n'a pas apporté à la vie quotidienne de chacun des changements aussi marqués que l'électricité, la voiture, le téléphone, le cinéma, l'avion. Dans ses travaux et ses jours, l'individu ne sent pas encore l'informatique, si ce n'est par des petits détails comme le fait qu'il reçoit plus vite et mieux tenu le relevé de son compte en banque. Si révolution il y a, ce n'est qu'au niveau socio-économique et scientifique.

L'ordinateur est en effet essentiellement un moyen d'action à l'échelle et pour les entreprises du *groupe*. Est-il révolutionnaire sur ce plan? Certes plus que sur le plan de la destinée personnelle. Mais j'observe qu'en économie, par exemple, l'ordinateur n'a pas encore suscité l'éclosion d'activités entièrement nouvelles (hormis bien entendu l'industrie qui le crée lui-même) comme le charbon a suscité la machine à vapeur. Car même dans ses applications à l'économie, il n'est pas un bien consommable ou un rouage de biens consommables. Il appartient plutôt à la catégorie des *outils*, ou plus généralement des *moyens intellectuels* permettant de mieux faire ce qu'on faisait, ou d'oser des entreprises qui sans lui paraissaient impossibles. L'informatique conditionne donc le progrès d'une manière *indirecte*, un peu comme l'imprimerie, ou le papier, ou la géométrie.

De quelle manière exactement? Je crois que c'est surtout en débouchant des goulots d'étranglement, c'est-à-dire en rendant possible l'accélération ou l'expansion de l'économie dans les secteurs qui menaçaient d'étouffer. Vous l'avez sûrement senti au cours de la troisième partie. Le mouvement des salaires, des comptes en banque, des assurances, des passagers d'avion, des marchandises devient si abondant et si complexe qu'il n'a été possible de tenir la course que grâce à l'ordinateur. C'est le cas de dire qu'il est arrivé à temps. De même on ne peut gagner sans ordinateur le pari d'administrer la médecine et l'enseignement, avec un registre et une efficacité considérablement accrus, à des masses énormément gonflées. De même encore, vous avez vu que la production intellectuelle ou juridique devenait beaucoup trop volumineuse pour qu'on puisse encore l'exploiter à la main.

Remarquez que la plupart des grandes inventions sont apparues ainsi à des moments où il fallait porter à un niveau supérieur d'efficacité des méthodes de travail, de pensée, de communication, qui craquaient sous la poussée de développements sociaux ou techniques à croissance exponentielle. L'imprimerie fut un exutoire nécessaire pour des échanges intellectuels qui s'étaient accrus au point qu'on ne pouvait plus les assurer à la plume. L'énergie du charbon, puis de l'électricité, puis du pétrole, permit d'animer les machines qui, à chaque époque, se pressaient aux portes pour multiplier les richesses. Quand la foule des gens occupés et le rythme de leurs occupations se furent accrus au point qu'il n'était plus concevable d'assurer leurs transports par carrosse, le train prit le relais et permit de grossir encore la foule et de presser encore le rythme.

Ce rôle des grandes inventions qui consiste à autoriser (donc aussi à promouvoir) l'accélération des progrès économiques est particulièrement frappant avec l'informatique, puisque sa vertu technique essentielle est la vitesse. Son aptitude à traiter logiquement de grandes masses de données ou symboles est certes essentielle elle aussi, mais elle ne servirait à rien si elle ne s'exerçait pas nettement plus vite que ne le

permettraient des procédures *homo*. Il est probable que déjà avec une multiplication de la vitesse *homo* par cent, on aurait rendu à l'économie des services très appréciables. Or, elle fut multipliée par dix millions! C'est l'ampleur fantastique de ce saut qui lui donna son allure révolutionnaire. Grâce à lui, l'économie devient à la fois *mieux assise* et *moins visqueuse*. On augmente de plusieurs fois la pertinence des entreprises, et on diminue de 100 fois leur inertie. Tous les rouages peuvent réagir beaucoup plus vite sur des bases plus solides. L'éditeur qui publie ce livre, et s'est mis sur ordinateur, ainsi que moi, l'auteur, nous n'aurons plus à attendre deux ans pour savoir s'il est bien ou mal accueilli. Nous suivrons sa diffusion à la semaine, et saurons même dans quels milieux il marche plus ou moins bien. Ce qui nous sera d'une aide inappréciable pour orienter nos prochains pas. Nous débitions jadis dans un milieu trouble, presque impénétrable, et grâce à l'ordinateur nous pouvons y promener nos phares à mesure que nous avançons. Si on compare l'économie à un être vivant, on peut dire que l'informatique lui donne une vue plus pénétrante, une connaissance plus complète et surtout des gestes plus vifs.

Notez que pour affiner ainsi ses sens et ses organes, l'humanité doit faire sur elle-même une réflexion profonde, qui est peut-être le plus salutaire et le plus bouleversant des effets de l'ordinateur. Il n'est pas d'exemple que son introduction dans une entreprise, en forçant celle-ci à repenser tous ses rouages, n'ait pas conduit à une organisation nettement plus rationnelle du travail, à l'éclatement de vieilles routines injustifiées et pesantes, à un climat, un cadre, des méthodes plus agréables pour les hommes. L'ordinateur nous oblige à dénouer tous les fils de nos travaux dont nous acceptions jusqu'ici la trame sans réfléchir.

Ce sont là des services qu'on peut qualifier de révolutionnaires, même quand on veut se garder de l'enflure journalistique. Quand il pousse à un diplodocus un nouvel œil plus perçant, une nouvelle mémoire plus sûre, et une nouvelle patte plus agile, on parle même de *mutation*. Mais là où l'adjectif « révolutionnaire » ne saurait être marchandé en aucune façon, c'est dans les applications de l'informatique à la science. A mon avis, c'est là que se situe l'essentiel de l'affaire. Car nous avons vu qu'avec l'ordinateur la science non seulement voit plus loin et conclut plus vite, mais encore renouvelle complètement son sang même. Tout change à partir du moment où on peut calculer les effets de toutes les lois qu'on sait écrire, ajuster avec l'aisance d'un danseur des modèles aussi subtils que l'imagination les suggère, exploiter des statistiques aussi fournies qu'on veut. Les barrières à l'examen scientifique, qu'on croyait infranchissables, sont renversées. Si les techniques et les idées s'accumulent aujourd'hui si vite, si l'audace des chercheurs n'a plus de borne, si les conquêtes se périment en 20 ans, c'est à cause de l'ordinateur. Vous ne le voyez pas et ne le sentez pas, car on ne vous parle que des résultats, mais ce qui les a rendus possibles c'est, dans le fond des laboratoires, le grand four électronique à données.

Et comme c'est la science qui en fin de compte façonne notre destin, si l'ordinateur y a porté la révolution, on peut bien dire qu'il l'a portée en tout

La vie va-t-elle se robotiser?

Oui mais — disent les esprits chagrins — à quel prix? L'informatique détruit le charme de la vie qui était en partie liée à son imprévu, et nous transforme tous en

robots, qui à leur tour agiront bientôt sous l'ordre d'ordinateurs robots. Et d'égrener toutes les complaintes ou satires que j'ai formulées en épigraphe de ce livre :

« *Bientôt notre cerveau deviendra inutile ; les ordinateurs penseront pour nous, et nous n'aurons plus qu'à digérer.* »
« *A quand le mariage par ordinateur?* »
« *L'homme va enfin réussir à se transformer en robot.* »
« *Bah! En quoi la conquête de la Lune peut-elle m'intéresser? C'est une affaire d'ordinateurs.* »
« *N'appelez plus votre médecin, appelez un IBM 360.* »

Ce sont là des prototypes de messianisme nébuleux, je regrette de devoir le dire à ceux de mes lecteurs qui s'en seraient laissé séduire. Ces énoncés ne se basent sur rien d'autre que des dérapages verbaux, comme celui qui fait dire elliptiquement « l'homme a appris à voler » pour signifier qu'il a appris à faire des machines qui volent et auxquelles il peut confier sa carcasse qui, elle, reste tout aussi rampante qu'à l'ère paléolithique. De la même façon, l'homme ne se programme pas lui-même parce qu'il programme un ordinateur, c'est plutôt l'inverse, il doit se surpasser; et la locomotive ni même l'auto n'ont transformé ses jambes en roues à pistons. Loin de se mécaniser, l'homme doté de machines s'humanise. C'est le paysan de La Bruyère qui était une machine, une machine à sang ployée sur le sol. Depuis qu'une faucheuse-batteuse se charge du blé, le paysan s'est redressé et contemple le ciel. Rappelez-vous qu'on a entendu les mêmes litanies à propos de toutes les grandes inventions. De graves docteurs épouvantés par Gutenberg assuraient que, du moment où les écrivains n'auraient plus à calligraphier leurs œuvres debout, et que leurs lecteurs ne verraient plus leur script authentique, tout style et toute âme allaient disparaître des écrits. Au contraire, les auteurs prirent des soins plus grands parce qu'ils se savaient davantage lus, les idées circulèrent plus loin et purent être débattues plus à fond, et l'humanité enrichit et embellit prodigieusement sa culture. Quand parut le train, ce fut le même concert sur la fin du contact direct entre le voyageur et la nature, et Vigny déplora dans un très beau poème qu'il n'éprouverait plus jamais :

... « l'espoir d'arriver tard dans un sauvage lieu » ...

Certes le train fit perdre quelques éléments pittoresques des anciens voyages (encore qu'il en apporta d'autres) et je ne prétends nullement que la machine ne change rien, ni ne puisse rien enlaidir. Je demande seulement qu'on en fasse le bilan complet. Si on ne voit plus guère aujourd'hui des chevaliers arrivant tard dans un sauvage lieu, on voit, par contre, des millions d'hommes du commun qui, bien qu'à l'heure, arrivent ravis dans des milliers de lieux enchanteurs qu'avant le train ils n'auraient jamais atteints. Au total, l'amour de la terre, de toute la terre, s'est immensément étendu dans l'humanité, dans toute l'humanité. Le cinéma et la TV vont stéréotyper les goûts, clament les nostalgistes, en oubliant qu'on propose tant de programmes variés tous les jours que chaque nuance des personnalités les plus étranges y trouve de quoi se satisfaire et se parfaire. Les habitants de Provins que décrit Balzac, confinés à cinq types de draps par an et un sermon par semaine, étaient beaucoup plus stéréotypés que les foules infiniment bigarrées, en esprit comme en allures, qui sillonnent le monde du cinéma et de la TV. En général, les inventions techniques ne tarissent pas mais, au contraire, multiplient et renforcent les ressorts qui permettent aux individus (qui le veulent et le peuvent) d'être plus divers et plus originaux. C'est

l'absence, et non l'abondance, des moyens de connaître et d'agir qui uniformise les hommes.

L'ordinateur confirme cette règle trop souvent méconnue. Il ne diffère en rien des autres conquêtes de la science. Comme elles, il élargit le champ du possible offert à l'homme et par là accentue, et non pas aliène, sa responsabilité. A chaque étape de ce long cortège de merveilles commençant avec la roue, et dont l'ordinateur n'est que la plus récente et non la dernière, l'homme, ayant acquis une nouvelle puissance doit, pour s'en servir, aménager son espace ancien et donc interroger plus à fond ses desseins et ses habitudes.

Dès le début de ce livre, où nous avons feint pendant quelques pages de nous marier par ordinateur, nous avons reconnu qu'en fait l'ordinateur nous laisserait parfaitement maîtres de nos unions et n'enlèverait rien à leur poésie. Rien n'empêcherait ceux ou celles qui tiennent au frisson (délicieux en effet) de la surprise, d'essayer les rencontres de hasard à côté des rencontres réfléchies que proposerait l'ordinateur. Lequel, rappelez-vous, appliquerait des normes fixées non par un démiurge arbitraire mais par ceux-là mêmes qui cherchent des conjoints à leur convenance. Ceux-ci pourraient d'ailleurs, s'ils le souhaitent, instruire l'ordinateur de laisser flotter sciemment, autour des traits qu'ils aimeraient rencontrer chez le partenaire, des marges suffisantes pour que jouent à la fois une dose de sélection délibérée et une dose de découverte. Autrement dit, si les couples demandaient vraiment conseil à un ordinateur pour mieux s'assortir, ils seraient obligés, afin qu'on pût donner à cet ordinateur un programme pertinent, de scruter leurs rêves avec plus de finesse, de songer à une gamme plus large de possibles, et, après avoir reçu avis de la machine, de combiner, tester, éprouver une plus grande variété d'options.

C'est d'ailleurs là que gît le vrai vice de ce service imaginaire : il exigerait de l'homme qu'il approfondisse la connaissance et la maîtrise de lui-même, et élargisse son registre affectif et intellectuel, jusqu'à des niveaux qui sont aujourd'hui utopiques. De sorte que l'aide de l'ordinateur à l'appairage des hommes et des femmes ne saurait en effet être, aujourd'hui, qu'une caricature. Mais du fait de notre primitivisme, non du fait de l'ordinateur. Dès lors que nous ne pouvons lui donner que des bases de choix sommaires et fragiles, les suggestions qu'il nous fera seront nécessairement ineptes, et il vaut mieux, en effet, laisser jouer la spontanéité des êtres. Mais même si un jour notre regard intérieur devenait assez pénétrant et notre contrôle sur nos actes assez souverain pour rendre possible une pré-sélection subtile de partenaires par calcul sur fiches suivi d'essais en vraie grandeur et re-calculs, cela n'éteindrait pas encore la genèse enchantée de l'amour. Elle serait seulement reportée sur des plages encore plus secrètes. On n'aurait pas chassé le merveilleux des affaires du cœur, on l'aurait repoussé de plusieurs crans vers la sophistication en laissant un programme nous assister pour les ouvertures simples du jeu.

Je n'ignore pas que c'est dans le domaine sentimental que l'insertion de l'informatique paraît la plus rebutante. Il n'en faut que plus d'efforts sur soi pour s'obliger à comprendre ce qu'elle signifierait réellement. Si, cher lecteur, vous avez bien voulu accomplir cet effort (et m'excuser de vous l'avoir infligé) vous en conclurez, j'en suis sûr, que l'ordinateur jouerait le rôle d'une sorte de téléscope-microscope psychologique, et vous vous refuserez comme moi à croire que le champ laissé à l'amour disparaîtrait parce qu'on saurait mieux sonder les âmes et qu'on leur donnerait de meilleures chances pour rencontrer leurs sœurs. Dans les conditions (encore hypothé-

tiques) où le recours à l'ordinateur serait valable, il ne dégraderait pas notre lyrisme intime, et nous supposerait plus complètement hommes que ne le sont aujourd'hui les partenaires de tant de couples qui se marient parce qu'ils se sont amusés ensemble une fois au bal. De même, nous avons vu qu'en médecine l'ordinateur n'empêchera aucun docteur intuitif d'exercer son intuition, mais qu'il y *ajoutera* un panorama de la pathologie bien plus complet et précis que n'en peut porter dans sa tête le plus grand médecin. Il faut donc faire infiniment plus de bonne médecine, et au plus près de l'humain, si on veut utiliser un ordinateur, que quand on se borne au docteur des familles. Et si demain maman demande à son ordinateur de ménage de déterminer automatiquement le nombre des petits pois à mettre autour du gigot, il faudra, pour programmer cette action, que des chercheurs aient poussé bien plus loin que ce ne fut jamais fait l'étude des relations gustatives et sociologiques dans le complexe gigot-petits pois-convives.

L'idée que l'ordinateur éteint le charme de la vie et nous transforme en robots n'est nulle part mieux démentie qu'en gestion. Il n'y a rien de beau, rien qui soit « authentiquement humain », dans ces cent dos qui, derrière les cent cubes de tôle du grand hall, se penchent 8 heures par jour devant dix mille tableaux de chiffres. L'ordinateur qui exécute les opérations à leur place, brillant et serein dans un coin du hall, porte un bien plus fier témoignage du génie de l'homme, et permet aux deux cents yeux délivrés des additions de se porter sur les bilans, c'est-à-dire sur des problèmes bien plus chargés d'humain. La plus grande fantaisie qu'on observe dans le coloriage des voitures est due à l'intervention d'un petit ordinateur qui peut enfin diriger les jets de peinture automatiquement, selon un programme aussi multiforme qu'on veut, voire aléatoire, sans qu'on ait à interrompre la chaîne pour remettre en route une autre teinte.

Un auteur qui croyait défendre « l'humain » contre le « mécanique » déclara : « Le problème n'est pas de savoir si l'ordinateur arrivera à penser comme un homme, mais de savoir si l'homme n'arrivera pas à penser comme un ordinateur. » Or un ordinateur ne pense pas, il exécute vite les ordres qu'il a reçus d'hommes qui ont pensé, et bien plus intensément que s'ils n'avaient pas eu à donner ces ordres. Qu'est-ce que notre auteur voulait donc dire? Rien de clair. A son insu peut-être, il jouait sur une association confuse d'images confuses : ordinateur égale mécanique égale sans âme, ça fait tac tac tac quand ça imprime ses résultats, l'homme va penser par tac tac tac. Dans cette veine on dirait aussi que les pensées de Socrate étaient laides, ou que le temps imite son baromètre. Ce glissement subreptice de l'apparence extérieure au mécanisme intérieur est l'un des procédés les plus vieux — et les plus vicieux — pour halluciner les esprits. La machine à écrire aussi fait tac tac tac. Va-t-on en conclure que l'écrivain qui s'en sert « pense comme elle »? Un jour, m'étonnant devant des jeunes qu'ils puissent « s'ennuyer » à l'époque passionnante où l'homme conquiert la lune, l'un d'eux me sortit l'une des phrases que j'ai mises en épigraphe : « En quoi ça peut-il nous intéresser? C'est une affaire d'ordinateurs. » Montrant une ignorance abyssale à la fois de l'ordinateur, de la lune et de l'homme. Je dus lui expliquer que, derrière chacun de ces boutons d'ordinateur qu'il méprisait, se tenaient Copernic, Tycho Brahé, Galilée, Newton, Laplace, des siècles de recherches et de luttes, la conjonction des deux plus hautes et plus poignantes curiosités de notre espèce, celle pour le cosmos et celle pour l'atome, des myriades de physiciens, chimistes, ingénieurs, sociologues et poètes mobilisant et bandant toutes les connaissances et toutes les

volontés, sans compter le courage prodigieux des astronautes et la pertinence fantastique de toute l'infrastructure qui les portait et dont la pièce la plus étonnante était le programme de ces ordinateurs qui, silencieusement, sans faillir un instant, donnaient le chemin à suivre en tous les points du vaste espace.

Jamais en vérité on n'avait pu voir de façon plus éclatante, plus émouvante, combien l'ordinateur est un instrument d'humanité, tout pétri d'elle et la magnifiant. Jamais il n'avait manifesté plus clairement son véritable rôle, qui est de mobiliser tout ce que les hommes savent et peuvent prévoir, au service d'un prochain pas. Voici en effet l'idée de base : on confie simplement à une machine toutes les besognes dont la **science acquise préalablement nous a rendus sûrs,** et dont il n'y aurait donc plus aucun intérêt à charger des humains. Ceux-ci restent ainsi plus libres et mieux armés pour la **science encore incertaine,** celle qui reste à faire. C'est la démarche même, quasiment la définition, du progrès. *Une machine n'est jamais que la mise sous boutons de puissances ou connaissances antérieurement accumulées et suffisamment domptées.* Sa fonction historique n'est jamais que de servir notre bras ou notre cerveau pour que nous accomplissions mieux notre tâche propre et noble d'hommes, qui est d'explorer des horizons nouveaux. A nous l'inconnu, à elle le connu. La particularité — notable — qu'apporte l'ordinateur dans cette ronde ancestrale de l'esprit avec ses instruments, c'est que, grâce à lui, et à la condition de bien prévoir son travail en réfléchissant très fort, les puissances et connaissances antérieures peuvent être mobilisées avec un rendement beaucoup plus grand qu'on ne le soupçonnait, et dans des domaines beaucoup plus nombreux; parmi lesquels apparaît enfin, comme inconnue déchiffrable, l'homme même.

Mais c'est précisément cette ouverture qui effraie.

Il faudra bien qu'un jour les hommes cessent d'avoir peur de se connaître.

Est-on menacé de chômage?

Cette interrogation angoissante se pose en vérité pour l'automation plutôt que pour l'ordinateur proprement dit. Examinons-la pour les deux à la fois.

Jusqu'ici l'incidence globale de l'ensemble ordinateur-automation sur le marché du travail a été faible. Dans les industries, c'est-à-dire plutôt du côté automation, bien des emplois anciens ont certes disparu, mais il en est apparu presque autant de nouveaux. D'abord parce que l'économie se diversifiait et s'étendait à mesure que l'automation améliorait, ou accélérait, les fabrications. Ensuite parce qu'elle-même donnait lieu à une industrie nouvelle, transformant ainsi certains des travailleurs qui martelaient des tôles en travailleurs qui montaient les savants rouages de machines martelant les tôles à leur place.

Le bilan montrait quand même une diminution du nombre total des heures de travail nécessaires. Dans les bureaux, c'est-à-dire du côté informatique, ce bilan a, jusqu'ici, montré plutôt une *augmentation* de l'emploi dû à l'ordinateur.

Il n'y a presque pas de cas où l'introduction de l'informatique ait conduit à économiser du personnel de gestion. On a cru longtemps que c'était là sa principale justification. Aujourd'hui, on a compris l'erreur. Le rôle primordial de l'informatique dans une organisation est **d'accroître sa vitesse de réaction.** En outre, elle améliore

la connaissance des faits, rend les décisions plus pertinentes, et soulage les employés d'une série de besognes assommantes. Mais du fait même que l'organisation tourne mieux, elle offre un champ d'action plus vaste, et au total emploie le même nombre de gens, mais à un plus grand nombre de tâches plus raffinées. Depuis que dans son arrière-boutique elle a mis un ordinateur, votre banque, à ses guichets, vous fournit plus vite une gamme plus étendue de services, avec le même nombre d'employés... peut-être plus souriants.

D'autre part, dans son arrière-boutique, l'ordinateur demande les soins de milliers de gens, 280 000 en France seulement, avons-nous vu. C'est beaucoup. Et une moitié de cet effectif est à nouveau composée de travailleurs en faux col.

Les études économiques confirment l'idée générale que l'ordinateur n'a pas, sur le chômage, un effet spécial, différent de celui qu'on attribue depuis longtemps à l'ensemble des améliorations technologiques. A savoir qu'elles provoquent incontestablement une diminution du temps de travail qu'on demande aux hommes, mais que cette diminution est lente, compensée par l'expansion continue des activités économiques. La courbe résultant du jeu de ces deux tendances contraires a une pente descendante assez faible pour laisser aux sociétés prévoyantes le temps de s'adapter. Là où peut-être l'ordinateur aura une incidence plus spécifique, c'est dans l'accélération du processus bien connu qui déplace de plus en plus la main-d'œuvre des secteurs primaire (agriculture) et secondaire (industrie) vers le tertiaire (bureaux et services). Notons que cette évolution va dans le sens d'une vie plus agréable.

Est-on menacé de tyrannie?

Le premier poème connu de la littérature égyptienne, datant de l'an 4000 avant J.-C., pleure... le bon vieux temps! Dès 1770, le grand écrivain et homme politique anglais Edmund Burke écrivait : « L'âge de la chevalerie n'est plus. Celui des sophistes, des économistes et des calculateurs lui a succédé, et la gloire de l'Europe est éteinte pour jamais! » Vous voyez que le « nostalgisme » est de tous les temps, et n'eut pas besoin d'ordinateurs pour fleurir. Il était indispensable de remettre en perspective l'ancestrale résistance au progrès lorsqu'on entend accuser l'ordinateur de nous transformer non seulement en robots, mais encore en esclaves. Mais bien que l'histoire démente ces complaintes aussi régulièrement qu'elles fusent, chacune d'elles mérite qu'on l'examine pour son compte; d'abord par devoir d'objectivité, ensuite parce que, un jour, elles peuvent devenir vraies, enfin parce que, même quand leurs conclusions se révèlent fausses, leur inspiration est respectable, puisqu'on y trouve le souci le plus haut et le plus juste que puisse avoir un homme réfléchi, le souci pour le charme de la vie et la liberté de l'individu. Voyons donc, sans parti pris de modernisme, s'il est vrai que l'informatique menace de nous tyranniser.

Cette menace peut se situer sur quatre plans distincts.

On peut d'abord craindre que l'homme devienne l'esclave de l'ordinateur dans le même sens où on le dit esclave de la machine ou de ses passions : il oublie de vivre pour son plaisir et ne songe qu'à mettre des ordinateurs partout et à préparer leur pâture. Que cette pente existe n'est pas niable. Mais elle ne surgit pas spécialement à propos de l'ordinateur. C'est la pente générale qui conduit l'homme à oublier ses

fins et adorer les moyens qu'il a institués pour y parvenir. Il y a certes une tendance moderne à faire de « l'informatique pour l'informatique », mais les gens les plus éloignés du machinisme ne font-ils pas aussi de « l'art pour l'art »? L'ordinateur n'est donc pas coupable en tant que tel, il attire simplement sur lui ce dérèglement ancestral de l'homme, comme l'avaient fait auparavant tant d'autres fruits de ses efforts. L'auto n'est pas responsable de ce que son conducteur assouvit à son volant des appétits de puissance et d'agressivité. Il faut, dans tous les cas où l'homme invente de nouveaux moyens d'action, qu'il apprenne à s'en servir en maître au lieu de vivre pour eux.

On peut encore concevoir la tyrannie de l'informatique comme un carcan qui assujettit à des règles étroites (et ennuyeuses) tous les actes de la vie, en particulier dans la vie économique. L'employé d'un bureau « informatisé » deviendrait alors le genre d'ilote dominé par une machine que Charlot a campé dans « Les Temps Modernes ». Cette satire est très amusante, mais me paraît déjà fausse dans l'industrie, où tous les développements modernes tendent — au moins en principe — à moduler les tâches des ouvriers. Il est vrai qu'il est plus facile de supprimer un geste pénible, exigeant un effort physique, que d'éliminer un travail fastidieux par sa monotonie et son caractère parcellaire. Des études récentes tentent justement d'humaniser le travail de l'ouvrier sur une chaîne, appelé O.S., ou « ouvrier spécialisé », alors qu'il n'a souvent en fait aucune spécialité et que son activité professionnelle se réduit encore à quelques gestes vite appris et répétés ensuite tout au long de la journée. Ces opérations quasi machinales sont précisément du ressort de l'automatisation. Mais dans le tertiaire, la satire serait tout à fait insoutenable. Je connais des dizaines d'installations d'informatique dans des bureaux de toutes sortes, et pas un seul exemple où elles n'aient eu pour résultat de rendre la vie des employés moins monotone, plus détendue, plus plaisante, puisque toutes les tâches mécanisables ont été justement confiées à l'ordinateur, et que c'étaient elles qui exigeaient la discipline la plus pesante. Au plus peut-on dire que l'informatique exige de chacun qu'il fasse mieux ce qu'il a à faire, mais on peut difficilement appeler « tyrannique » l'obligation de crayonner la case idoine des formulaires. Le charme consisterait-il à crayonner la case d'à côté? Si c'était vrai, ça ne serait certainement pas pour le geste même, puisqu'il ne change pas. Alors, le charme serait-il lié à l'erreur? Faisons attention que, de fil en aiguille, cela nous mènerait à ne tenir pour charmante que la vie préhistorique. Non, direz-vous, le charme est lié au droit de faire des erreurs. Ne vous inquiétez pas, ce droit vous restera, car l'ordinateur ne fera que repousser votre marge de fantaisie vers des zones de plus haute intellectualité, où il sera plus digne de vous, et plus intéressant, de vous tromper. D'ailleurs, au prochain pas, avec un développement suffisant des capteurs de données, vous n'aurez même plus à crayonner des cases, sauf quand cela demandera réflexion, et alors ce sera amusant.

Une forme plus abstraite, mais plus grave, de la tyrannie des ordinateurs, toujours dans le domaine de la gestion, consisterait en ce qu'ils accentuent la pyramide de l'autorité. Pendant longtemps on a cru que l'ordinateur installé à tous les carrefours de la vie des grandes entreprises allait rendre inutiles les organes de direction dispersés et successifs, et permettre à un seul, ou à un petit groupe, au sommet, de réunir tous les fils du commandement. L'expérience, là encore, a montré que l'inverse était vrai. Il s'est avéré que de laisser tous les problèmes s'accumuler jusqu'au niveau suprême coûte encore plus cher quand ils sont traités par un ordinateur que quand ils

sont traités par des hommes. Si on laisse le flot des informations s'enfler à travers tous les étages d'une grande société jusqu'à ce qu'il parvienne, chargé de tout à la fois, à un super-patron unique, son traitement en un seul coup par un ordinateur sera infiniment plus lourd que la somme des traitements des problèmes séparés. On a donc intérêt à organiser le circuit informatique de façon à clore le maximum possible de traitements automatisés le plus tôt possible, ne transmettant que leurs résultats élaborés aux échelons supérieurs. Ceux-ci ne connaissent alors que des problèmes dont les éléments sont les solutions des problèmes précédents. On évite ainsi le gonflement exponentiel de l'information, et donc du prix de son traitement, à mesure qu'elle escalade la hiérarchie. Il y a plus. Ces conclusions partielles qu'on débite à un échelon intermédiaire sont plus complètes, plus solides, plus fraîches qu'avant l'informatique. Donc il y a moins de risque à déléguer une plus grande part de responsabilité à cet échelon. Au total, l'expérience a démenti la crainte des débuts, qui provenait d'un réflexe irréfléchi : parce que l'ordinateur a l'air d'un gros monolithe qui avale tout, on en a déduit que sa présence allait transformer l'entreprise en un gros monolithe où un seul pense pour tous. En fait, il donne une forte prime à la décentralisation, c'est-à-dire au reflux d'un maximum de décisions vers la base.

Une troisième tyrannie qu'on peut craindre concerne l'ordre de l'esprit. Beaucoup de gens croient que les résultats issus d'un ordinateur sont tellement certains qu'ils s'imposeront comme des décrets indiscutables. D'où le spectre d'une orthodoxie particulièrement rigide parce que émanée d'une machine sans âme. L'auteur que j'ai évoqué plus haut a résumé ce phantasme dans la phrase suivante : « Il n'y a pas de problème pour l'ordinateur. »

Quelle profonde, ahurissante méprise! Le fait que l'ordinateur permet de pousser les études plus loin et de répondre à plus de questions n'implique nullement que les réponses soient plus sûres. Elles n'ont pas un iota de certitude de plus que les hypothèses et les données admises par l'analyste qui a tracé le programme du traitement. Disons au mieux que l'ordinateur nous rend sûrs des *calculs*, et permet d'essayer une plus grande gamme d'hypothèses et de données, ce qui augmente les chances de trouver les bonnes. Mais elles devront toujours en fin de compte prouver leur vérité, comme chez nos grands-pères, par leur accord avec l'expérience, lequel accord reste aussi ouvert que toujours au débat. Que dis-je, le débat se vivifie. Car l'obligation de programmer minutieusement les hypothèses, les données et les lois qu'on applique, *les détache plus nettement*. Et si le programmeur veut tâtonner, eh bien, l'ordinateur manifestera ces tâtonnements. L'ordinateur n'établit aucune orthodoxie, il étend simplement le terrain que peut couvrir la spéculation, et le déblaie d'une masse de scories; ce qui, loin d'interdire la critique, la conduit mieux vers l'essentiel. D'ailleurs beaucoup de programmes contiennent des ordres spécialement prévus pour faire sortir des résultats intermédiaires, de façon qu'on puisse juger les conséquences des hypothèses, lois et données retenues, non seulement sur le thème principal et final, mais encore sur des aspects adventices, et même d'après le chemin suivi. En bref, les seuls problèmes que l'ordinateur n'a pas sont ceux de l'*exécution* des traitements (et encore, ce n'est vrai qu'après mille tourments chez les programmeurs). D'où résulte que l'examen se reporte sur les *tenants et aboutissants*, qui sont les vrais problèmes.

C'est bien ce qu'on constate dans la vie scientifique et technique moderne. Elle n'a jamais autant scruté les phénomènes, disséqué les théories, discuté les affirmations

que depuis que l'ordinateur les met à nu et en relief, entourés de toutes leurs implications.

Supposez que demain, grâce à un ordinateur, votre agence de tourisme, au lieu de vous laisser patauger parmi des renseignements vagues sur les stations de vacances, déploie devant vous leur description enfin exacte, fine et complète, avec les heures et les aires de l'ensoleillement, la proximité et la nature des plages ou forêts, la transparence de l'eau ou les pentes des sentiers, etc. Direz-vous que l'ordinateur de votre agence exerce sur vous une dictature spirituelle? Ne faudrait-il tenir pour libres que les hommes non informés?

La quatrième et dernière acception possible de la « tyrannie de l'ordinateur » est la plus sérieuse, et serait de loin la plus grave si elle était juste. Elle concerne l'emploi de l'ordinateur comme *fichier général automatisé de l'ensemble des citoyens.* Vous n'ignorez pas que, dans les États modernes super administrants, de nombreuses données sur vous peuplent de nombreux bureaux : votre état civil avec mariages, divorces, enfants, vos résidences avec adresses, valeur, surfaces corrigées, votre voiture, votre bateau, vos diplômes, vos maladies, vos revenus, les époques où vous avez demandé un passeport, vos réservations d'avion, votre patente, vos brevets et licences, votre casier judiciaire et — s'il n'est pas blanc — vos délits et procès, les condamnations que vous avez subies, assorties de leurs motifs, etc. Jusqu'ici, il n'était pas facile, même à la police, de collecter tous ces renseignements sur un individu, car ils étaient couchés selon des formats fantaisistes sur de méchants papiers ou cartons dispersés dans des dizaines d'administrations différentes. C'était aussi la source de nombreuses difficultés de transmission entre l'Administration et les citoyens; par exemple, la Sécurité sociale dépense 2 % de son budget pour corriger des erreurs de transcription ou retrouver des assujettis qui ont changé d'adresse.

Or l'ordinateur permet d'éviter tous ces tracas. On peut rassembler la totalité de ces renseignements, en code, sur une seule fiche pour chaque citoyen, l'ensemble de ces fiches étant stocké dans une mémoire de masse. Ou dans trois grands fichiers, par exemple : civil, médical et socio-économique. Il y faudrait, pour un pays comme la France, mettons, dix milliards de mots, ce qui est parfaitement concevable. Les choses seraient organisées de telle manière que chaque fois que vous avez un échange juridique avec un guichet de l'État, celui-ci, en tapant à la machine l'information afférente à votre acte, envoie automatiquement sa traduction en code à votre fiche, selon le grand principe de l'informatique qui veut qu'on recueille et absorbe les données au plus près de leur source. Ainsi le GFNO (Grand Fichier national *Ordino*) serait perpétuellement tenu à jour. Et chaque fois qu'une administration aurait affaire avec vous, elle téléphonerait au GFNO enserré dans quelque voûte sous un massif des Alpes, et recevrait par téléscripteur toute votre fiche. Pendant qu'on y est, on monterait aussi un GFNO pour les entreprises. Et l'État saurait tout sur chaque citoyen ou chaque société en un instant (mettons quand même une heure, vu que le fichier à faire défiler est énorme même dans l'optique *ordino*). Remarquons qu'en l'état actuel de l'art, une telle réalisation serait *extrêmement difficile*. Mais supposons quand même qu'elle soit faite, et examinons ses conséquences.

Elle incarnerait bien les prophéties des auteurs de Science-Fiction apocalyptique qui nous annoncent que bientôt nous serons tous réduits à l'état d'électrons matriculés circulant dans une Métropolis tentaculaire sous le contrôle absolu d'un Bureau politique qui, grâce à un ordinateur géant, sera omniscient et omnipotent.

Observons d'abord que ***si*** les pouvoirs maîtres du GFNO n'étaient pas méchants, ils pourraient s'en servir pour le bonheur des hommes, et ce d'autant plus qu'il contiendra sur eux des renseignements plus caractéristiques. Car ***si*** les pouvoirs maîtres du GFNO ne le consultaient que pour faire le bien, ils pourraient d'une part augmenter dans un rapport considérable la qualité de l'administration, d'autre part tracer des plans beaucoup plus pertinents, enfin et surtout, apporter aux sciences de l'homme, de l'économie et de la société, un trésor inégalable de données, permettant des éclairages et des études de la plus grande conséquence. Vous admettrez donc que, dans l'hypothèse d'un gouvernement pur et bon, il serait très bénéfique de mettre sur pied un GFNO.

Mais aucun gouvernement n'est pur et bon, et Montesquieu assurait que « tout homme ayant du pouvoir est porté à en abuser ». Le GFNO donnerait cette tentation au gouvernement cent fois par jour. Il pourrait persécuter à loisir ses ennemis en déterrant sur leur compte tous ces détails qui, habilement gonflés par la polémique, peuvent faire paraître coupable n'importe quel innocent. Il pourrait détecter les moindres écarts fiscaux et traquer leurs auteurs avec une efficacité irrésistible. Il jouerait sur les maladies, les revenus, les goûts des citoyens pour favoriser les uns ou brimer les autres. Il pourrait à son gré enfermer les uns dans leur passé ou en libérer d'autres en maintenant ou supprimant sur leurs fiches la mention d'actes délictueux même après que l'individu s'en soit rédimé. Il pourrait ressusciter discrètement les fameux « livrets de travail » contre lequel les ouvriers ont tant lutté. Il siégerait pratiquement aux conseils d'administration de toutes les sociétés. Et tous ces pouvoirs d'intervention, il en aurait le privilège exclusif, puisqu'il serait seul détenteur du GFNO.

Cette affreuse perspective peut parfaitement devenir réalité. Mais réfléchissons plus à fond. Dès aujourd'hui, et sans ordinateur, la plupart des renseignements sur les personnes physiques et morales *existent*, nous l'avons vu. Mais ils existent sous des formes inefficaces, ce qui permet d'espérer que les « cabinets noirs » n'auront pas le courage de les mobiliser. C'est une consolation illusoire quand il s'agit de gouvernements totalitaires, qui n'hésitent pas à dépenser tout le temps et tout l'argent qu'il faut pour remplir leurs terribles dossiers. Ces gouvernements-là n'ont pas attendu l'ordinateur pour mettre en fiches leurs sujets, corps et âmes. Donc l'État en connaît d'ores et déjà assez sur les citoyens pour les dominer de façon totalitaire, **s'il le veut,** sans ordinateur. D'ailleurs, même dans les régimes qui respectent l'indépendance des individus, l'État a en mains tous les moyens de ne pas le faire *occasionnellement* si l'envie lui en vient. La facilité plus grande de déroger *systématiquement* que lui donnera le GFNO ne multipliera-t-elle pas ces envies, l'occasion faisant le larron? Certes, mais cela n'empêche que le risque vient plus du climat politique dans lequel baigne le gouvernement que de la plus ou moins grande perfection de ses instruments. L'antiquité déjà connut plusieurs tyrannies d'État sur les vies privées, et l'Inquisition précéda de beaucoup l'ordinateur.

Et puis il y a quelque chose d'anormal, de dissonnant dans l'idée qu'il faut que l'État tourne mal pour que les citoyens soient libres. Faire d'une imperfection volontaire dans l'une des parties du corps social le seul garant de la liberté, c'est supposer celle-ci bien désemparée, et la mettre bien bas. Quel aveu d'impuissance de la part des citoyens que de demander leur protection à l'impuissance de leur État! A ce compte-là, il aurait aussi fallu refuser à l'État l'usage du télégraphe et du téléphone, pour ne pas parler de la lance à incendie contre les manifestations, qui ont augmenté sa

puissance, et très précisément la suprématie de son appareil sur l'individu, plus encore sans doute que ne le ferait un GFNO.

Le grand problème moderne, d'exploiter la technique pour l'émancipation et non pour l'asservissement des êtres, doit se résoudre non pas en tournant le dos à la technique, mais en éduquant ceux qui l'exploitent. Or l'informatique est d'une versatilité telle qu'on peut l'employer de toutes les manières qu'on veut. Rien n'empêche, il est au contraire facile, d'agencer un GFNO de façon que ne puissent y figurer et persister que des éléments d'information bien déterminés. Il suffit pour cela de coder tous ceux qu'on admet, et de munir le programme d'une instruction qui fait refuser à la machine tout ce qui ne porte pas ce code. Et pour être sûr que les usagers du GFNO ne donneraient pas indûment aux informations illicites le code des informations licites, on peut confier le codage à un corps d'agents assermentés ne dépendant que du Parlement et qui ont seuls la « clé » *ordino* permettant l'accès à la machine. Ou encore on mettrait en place un programme de pré-examen qui ferait lire à l'ordinateur *le contenu même* de toutes les informations présentées, vérifierait si elles sont conformes au catalogue des informations admises, et rejetterait les *autres*. On peut aussi inscrire en programme des ordres pour effacer automatiquement les informations relatives à des actes sur lesquelles aurait joué une prescription. On peut encore arranger les choses de façon que seuls aient accès à l'ordinateur des gens qualifiés par un chiffre, ou que les tribunaux judiciaires aient la faculté de bloquer la sortie de certaines informations.

Bref, le fait même que le Grand Fichier national serait sur ordinateur, permettrait d'imposer à l'information et à son cheminement toutes les bornes, toutes les conditions qu'il plairait au corps social d'instituer. Certes, il appartiendrait à celui-ci de les définir exactement, ce qui ne serait pas une mince affaire, et l'obligerait à se pencher avec soin sur ce qu'il est bon et ce qu'il n'est pas bon qu'on sache à propos des individus. Mais n'est-ce pas le rôle fondamental des citoyens que de tracer au gouvernement les limites de ses droits sur eux ? Et, chose capitale, une fois ces limites clairement tracées, *les citoyens pourront être sûrs qu'elles seront respectées*, car ce n'est pas l'ordinateur, esclave rigoureux qu'il est de ses consignes, qui les transgressera. Alors qu'aujourd'hui, avec les informations volant de main en main sur des papiers, elles sont stockées, groupées, ventilées, exploitées au gré de fonctionnaires anonymes dans des officines impénétrables.

Ainsi le choix n'est pas entre l'individu débridé et l'individu fiché, il est entre l'individu fiché clandestinement et l'individu fiché ouvertement. On ne troque pas l'absence de contrôle contre un espionnage omniprésent, on troque un contrôle arbitraire contre un contrôle contrôlé. Si le corps civique s'élève à la hauteur des moyens que l'ordinateur lui donne, celui-ci se placera dans le droit fil de tous les progrès sociaux depuis 5 000 ans : il nous fera passer un peu plus du règne des barons à celui de la loi.

Résumons. L'ordinateur exécute plus correctement que des hommes les lois que les hommes lui dictent, à condition qu'ils les lui dictent bien. Donc, une fois de plus, il les met en face de responsabilités accrues. A eux d'en être dignes, ici comme devant toutes les inventions. Le corps civique ne saurait sans honte dire à ses savants : cessez de penser, car je suis trop paresseux ou trop timoré pour vous suivre. Le Grand Fichier national sur ordinateur peut être en effet un instrument pour dominer des citoyens paresseux et timorés, mais avec des citoyens alertes et courageux, il peut être une occasion remarquable pour mieux définir et mieux garantir leurs droits.

L'ordinateur ravira-t-il à l'homme la couronne de l'intelligence?

J'ai laissé pour la fin l'interrogation la plus lancinante que soulève la puissance des ordinateurs dans un domaine jusqu'ici réservé à l'esprit : ne peuvent-ils pas, dans un premier temps, devenir aussi intelligents que nous, et ensuite davantage?

A première vue, tout le présent livre tend à nous faire répondre emphatiquement *non*. Car je n'ai cessé de mettre l'accent sur la bêtise de l'ordinateur, incapable d'accomplir un pas que son programmeur ne l'en ait expressément instruit. Cela est très vrai, et cependant vous allez voir que cela n'empêche pas le programmeur d'instruire la machine *de façon qu'elle devienne moins bête* et présente finalement certains traits de l'intelligence. Afin de comprendre cet apparent paradoxe, examinons les programmes qu'on a conçus pour enseigner à un ordinateur à reconnaître des formes, mettons celles des lettres de l'alphabet, ou des chiffres écrits.

Devant la lettre à reconnaître, tracée en lignes noires sur un papier blanc, on place une « rétine » formée d'un grand nombre de minuscules cellules photoélectriques dont chacune vise une toute petite plage du dessin. Mettons, par exemple, qu'on décompose la figure présentée en 10 000 de ces menues plages, regardées chacune par l'une des 10 000 cellules. A la sortie de chaque cellule apparaît un signal 0 ou un signal 1 selon que la plage qu'il regarde est blanche, c'est-à-dire non traversée par un trait, ou noire, c'est-à-dire occupée par du trait. Je sais quel endroit répond par 0 ou par 1 d'après l'adresse de la cellule qui livre 0 ou 1. Ces 10 000 signaux sont les équivalents des influx nerveux qu'un œil humain regardant le même dessin aurait engendrés et envoyés au cerveau. Et on peut programmer l'ordinateur de façon qu'il reconnaisse les points situés sur des tracés « horizontaux » (c'est quand leurs tops sont accompagnés de tops dans les plages à gauche et à droite) ou « verticaux » (quand il y a aussi des tops dans les plages au-dessus et au-dessous). On peut même faire enregistrer à la mémoire les courbures des lignes, leurs points terminus, etc. Nous supposerons donc que nous avons instruit l'ordinateur d'extraire des tops et tips, et d'emmagasiner, toute cette information *descriptive* qui caractérise les contours du dessin qu'on lui demande de reconnaître.

D'autre part, un analyste peut avoir étudié et noté des **propriétés** qui distinguent, par exemple le dessin d'un *b* de celui d'un *m*. Pas, bien sûr, des propriétés globales telles que jambages ou rondeurs, qui ne parlent qu'à notre œil d'hommes. L'analyste doit découvrir des propriétés qu'on pourra tester dans l'amas des tops et tips recueillis par le capteur. Par exemple, combien le dessin a-t-il de terminus, en combien de points intersecte-t-il des horizontales, ou des verticales, ou les barres plus ou moins zigzagantes d'un masque, ou des rayons tournant autour de tels ou tels pôles. On ne se contente d'ailleurs pas d'explorer le dessin passivement. On y ajoute ce qu'on appelle des « *transformations filtrantes* ». Par exemple, un chaînon de transformation T_1 supprime tous les points qui sont agglutinés en pâtés et ne laisse subsister que ceux qui se trouvent sur les bords les plus lointains. Puis un chaînon de transformation T_2 ne laisse subsister de ces bords que les points anguleux où convergent deux parties courbes de directions différentes, ou encore les points les plus au nord, au sud, à l'est et à l'ouest. A la fin, un comptage C fait inscrire le nombre Z des points ainsi filtrés. Une suite de ces transformations en chaînons telles que T_1, T_2, C, se terminant par un bilan chiffré du résidu, ressemble un peu à des tamisages successifs, ou à des applications successives de réactifs chimiques ou de traitements thermiques, et

s'appelle une « *épreuve composite* ». Les analystes ont inventé de nombreux types d'épreuves composites capables de révéler, par le seul nombre qui émerge d'elles à la fin, bien des traits typiques secrètement tapis dans les circonvolutions d'un dessin.

Des données de ce genre, vous voyez tout de suite qu'un programme pourra instruire l'ordinateur de les chercher dans son amas de tops-tips, de les isoler et de les stocker. Vous comprenez aussi d'emblée qu'une fois les propriétés correspondantes dégagées, on puisse les **combiner** de manière à donner plus ou moins de **poids** à telle série de tops ou telle série de tips, par exemple : j'associe un 1 ou un 0 à toutes les explorations qui ont donné tel résultat, puis j'ajoute tous les 1. Je prends aussi comme poids les résultats des « épreuves composites ». Ces *combinaisons* effectuées sur les résultats des divers types d'analyses passives ou actives aboutissent à ceci que toutes les propriétés du dessin sont maintenant contenues dans une simple séquence de nombres se suivant en bon ordre. Je vais l'appeler l' « **étiquette d'identité** » du dessin, car elle résume l'ensemble des résultats des essais auxquels il fut soumis, et par suite « l'identifie ». C'est de la même façon qu'on arrive à enfermer les traits typiques d'une empreinte digitale dans une séquence d'indices numériques, par exemple : « 12, 431, 27, 296, 3, 17, 2, 120, 75, etc. », dont les spécialistes savent ce que chacun signifie en tant qu'indicatif du résultat de telle ou telle manière d'explorer ou transformer les contours tortueux. Une telle séquence est l' « *étiquette d'identité* » d'une empreinte digitale dans une méthode d'analyse déterminée.

Maintenant que l'ordinateur a réduit et condensé tout ce qu'il y avait de caractéristique dans le dessin présenté, et l'a traduit par cette chose simple qu'est une « *étiquette d'identité* », vous sentez poindre la suite de l'histoire. Dans une période antérieure, on aura présenté à l'ordinateur les lettres de l'alphabet ***en les lui notifiant***, et on lui aura fait déposer en mémoire, à côté de leurs codes, les *étiquettes d'identité* qu'il aura établies en opérant sur les dessins de ces lettres *exactement les mêmes explorations, transformations et combinaisons que celles qu'il va effectuer ensuite sur toute lettre inconnue qu'on lui présente.* C'est en ce point que l'analyste pense et raffine sa méthode. Il doit en effet dégager des modes d'exploration et de transformation, ainsi que des combinaisons numériques de leurs résultats, tels que les étiquettes d'identité d'une même lettre dessinée de nombreuses manières différentes soient, sinon tout à fait identiques, du moins très proches les unes des autres. C'est la partie conceptuellement la plus difficile de l'affaire, mais elle revient à l'homme. Supposant ce problème résolu, et la mémoire de l'ordinateur pourvue des étiquettes d'identité non ambiguës de toutes les lettres, avec leurs codes associés, le prochain pas est évident : quand l'ordinateur a formé l'étiquette d'identité d'un dessin de lettre qu'on lui propose et qu'il ne connaît pas encore, le programme la lui fait *comparer* à tour de rôle avec les étiquettes d'identité de toutes les lettres enregistrées dans sa mémoire et que l'ordinateur a déjà « apprises », c'est-à-dire auxquelles il sait assigner un code; et il déclare que la lettre inconnue est la lettre de son alphabet enregistré dont l'étiquette d'identité *se rapproche le plus* de celle qu'il vient d'élaborer.

Les mathématiciens ont mis au point de nombreux critères permettant de mesurer la ressemblance de deux séquences de nombres, et c'est pour pouvoir puiser dans ce fonds qu'on a traduit les dessins par des séquences de nombres. Il arrive néanmoins qu'avec la méthode d'analyse retenue, la comparaison faite selon ce critère entre une étiquette relevée sur dessin inconnu et les étiquettes enregistrées de dessins connus, laisse des doutes quant à l'identité du premier. Cela tient à l'immense variété dans les

manières de calligraphier. On organise alors une seconde étape d'apprentissage par la machine. On lui présente un très grand nombre de calligraphies variées, on lui laisse commettre ses confusions, mais on les lui fait noter. On lui fait ensuite compter combien de fois chacune d'elles s'est produite, puis combien de fois elles se reproduisent avec les mêmes tests mais où on a modifié les poids dans les combinaisons qui conduisent aux divers indices formant les étiquettes. Les combinaisons ainsi retouchées qui entraînent le plus grand nombre de succès sont donc plus efficaces, plus révélatrices que les autres, et l'ordinateur a su se les noter. On divise dès lors le programme en deux phases, une première de préexamen qui fait apparaître la *gamme* des combinaisons suscitées par le dessin, et une seconde qui, dans cette gamme, sélectionne les combinaisons que la machine sait désormais être les meilleures. Elle court-circuite tout le reste et n'effectue plus la reconnaissance qu'avec celles-là, ce qui augmente à la fois la vitesse de ses tests et la pertinence de son diagnostic.

Vous pensez bien que les chercheurs ont inventé d'innombrables manières de programmer sur ordinateur l'analyse des dessins et l'apprentissage de leur signification. Toutes suivent *grosso modo* les principes que je viens de vous esquisser. En les poussant plus loin et plus à fond, on parvient à programmer un ordinateur pour lui apprendre à classer de lui-même en catégories, des formes dont on ne lui a pas préalablement montré des prototypes. Puis, l'ardeur des informaticiens n'ayant pas de bornes, on a pu aller plus loin, et écrire des programmes qui permettent à un ordinateur de distinguer de quel côté d'un trait séparateur on a dessiné un triangle et de quel côté un carré, puis de trouver la démonstration d'un théorème élémentaire de mathématiques, puis de jouer quelques coups aux dames ou aux échecs.

Je sais ce que vous allez me dire. Que ces performances ont beau être étonnantes et spectaculaires, elles ne résultent quand même, elles aussi, que de l'application d'un programme qu'il a fallu que des hommes inventent et prescrivent à l'ordinateur. C'est vrai. Seulement, vous venez d'acquérir cette notion nouvelle et capitale qu'*on peut écrire des programmes qui permettent à un ordinateur* ***d'apprendre*** et de sécréter par lui-même, au terme de l'apprentissage, de nouveaux programmes pour mettre en œuvre très vite les aptitudes qu'il a acquises chaque fois qu'on lui propose une tâche nouvelle dans le domaine où il fut dressé. Après tout, n'est-ce pas de la même façon qu'apprend un enfant? A lui aussi il faut qu'un maître « montre les lettres » en lui disant « ceci est un *A*, ceci également, ceci est un *B*, etc. ». Après quoi le maître induit (autant qu'il le peut) l'enfant à effectuer sur des dessins, mentalement, comme l'ordinateur, des analyses, des épreuves, des combinaisons capables de dégager leur parenté avec les dessins patrons dont il a empreint sa mémoire. Et le mécanisme fondamental qui assure le progrès de l'apprentissage est semblable dans les deux cas : c'est le renforcement des opérations qui ont réussi et l'élimination de celles qui ont échoué.

Certes, l'apprentissage chez l'homme, bien qu'il faille *au début* l'organiser sur un schéma analogue à celui employé pour un ordinateur, conduit très vite à des aptitudes dont aucun programme ne saurait aujourd'hui rêver de reproduire la millième partie du début d'une. Le mieux qu'on ait pu faire jusqu'ici avec l'ordinateur est de lui faire lire des caractères d'imprimerie (à un rythme d'ailleurs plus rapide que celui de l'œil humain). Cela donne lieu, comme vous l'avez vu, à des applications importantes. Mais aucun programme n'a encore réussi à faire lire proprement à un ordinateur les écritures manuelles aux innombrables contorsions fantaisistes, lesquelles pourtant

ne résistent pas six mois au cerveau d'un enfant de sept ans. Impossible, malgré 15 ans de travaux acharnés, de faire identifier automatiquement à un ordinateur (précédé d'un microphone) les sons émis par un locuteur, et dont la reconnaissance dépend du contexte. Impossible de lui faire jouer mieux que quelques fins de parties d'échecs infiniment simples, à trois coups au plus. Et *a fortiori*, est-il besoin de le dire, impossible de lui faire mener une discussion, composer une symphonie ou exprimer des sentiments.

C'est qu'une fois les premières fondations dressées, l'esprit humain devient capable de se livrer à des élaborations d'une telle subtilité que ce même esprit, chez les plus grands psychologues, est incapable de démêler leur genèse. En particulier, la mémoire humaine possède des propriétés stupéfiantes, qui n'ont aucun pendant parmi les mémoires *ordino*. Elle n'emmagasine pas en binaire, c'est-à-dire ne se limite pas à des combinaisons de « tout ou rien », mais donne des nuances aux éléments qu'elle capte, c'est-à-dire une série continue de « valeurs d'intensité ». Elle les range dans des zones obscures et des zones claires, et les rappelle non par leurs adresses mais au moyen d'un très étonnant et encore très mystérieux mélange de balayages systématiques à travers des thèmes et de flashes sur des éléments isolés qui ont l'air de sauter du coq à l'âne. Combien de fois ne vous est-il pas arrivé de chercher un nom oublié en présence d'un ami, et quand celui-ci vous dit « le nom ne commence-t-il pas par un « *b* »? », vous claquez des doigts et dites « Ça y est, c'est Albert ». Nous sommes très très loin de comprendre ces mécanismes, et les comprendrions-nous que nous serions incapables de les reproduire avec nos techniques actuelles, ni avec celles dont on peut raisonnablement imaginer qu'elles surgiront dans l'avenir prévisible.

Nous rencontrons ici le véritable nœud du problème : c'est qu'au fond *nous ne savons pas du tout ce que c'est que l'intelligence humaine*. Et dès lors, il est tout simplement impossible de dire si nous pourrons ou ne pourrons pas en donner une semblable aux ordinateurs. C'est pourquoi les dissertations fines où tant de gens se complaisent à ce sujet manquent de fondation. Si on veut rester positif, force est de se contenter de vues plus modestes. Nous pouvons seulement noter et décrire du dehors, phénoménologiquement, le genre d'activités communément considérées comme tenant à l'intelligence, telles que la rédaction d'une dissertation. Et nous pouvons prévoir que des activités *de ce genre* ne seront pas accomplies par les ordinateurs dans l'avenir prévisible. Je me hasarderai même, quant à moi, à dire un peu plus. Le secret de la prodigieuse capacité du cerveau humain, c'est que chez lui la cellule élémentaire qui traite l'information n'a que la taille d'*une seule* (grosse) molécule, et que son travail est surtout de nature *chimique*. Or les plus petites cellules qu'on puisse imaginer aujourd'hui faire travailler dans les ordinateurs sont des grains de cristal, dont chacun contient des millions de milliards de molécules, et elles mettent en œuvre des échanges *physiques*, c'est-à-dire macroscopiques. S'il fallait assembler avec de telles cellules l'équivalent des 200 ou 300 milliards de neurones que contient un cerveau humain (notez que la fourmi en a 350 et la guêpe 800), et des mille milliards de synapses qui les relient les uns aux autres en tous sens, il faudrait remplir de circuits intégrés une centaine d'immeubles; et là-dedans, les opérations ne se dérouleraient encore qu'à la queue leu leu, une à la fois, au lieu de foisonner simultanément et explosivement en tout sens, comme elles le font dans notre cerveau. Ce qui permet à ce dernier, non seulement d'acquérir l'intelligence, mais encore, même sur le plan de la vitesse, de battre l'ordinateur chaque fois qu'il s'agit d'appréhender globalement des configu-

rations composées de myriades de gouttes. Nous avons vu que l'ordinateur doit les parcourir une à une pour pouvoir interpréter leur ensemble, alors que l'esprit saisit cet ensemble d'un seul coup d'œil. Il n'est en vérité si lent par rapport à l'ordinateur que lorsque lui aussi ne peut agir que séquentiellement, et c'est le cas quand il doit calculer.

On peut conclure qu'aucun programme, aussi colossal et génial soit-il, ne pourra sans doute conférer à un ordinateur la prodigieuse gamme d'aptitudes qui forme l'intelligence humaine, tant que cet ordinateur fonctionnera avec des éléments et des phénomènes **physiques**. Voilà de quoi nous rendre modestes et envisager avec calme les événements qu'il nous intéresse de supputer, c'est-à-dire ceux du prochain siècle. Mais il est impossible d'affirmer que l'homme ne réalisera pas plus tard, un jour très futur, des ordinateurs dont les cellules élémentaires seront des molécules, comme dans notre cerveau, c'est-à-dire des ordinateurs *chimiques et biologiques*. Ce jour-là on sera, peut-être, parvenu aussi à savoir comment fonctionne notre intelligence. Et alors elle pourra, peut-être, devenir l'enjeu de la course entre nous et nos machines.

Mais de telles machines ne seront-elles pas au fond... d'autres êtres vivants semblables à nous-mêmes? Auront-elles une âme? Qu'est-ce que l'âme? Impossible de répondre dans l'état actuel de la science. Laquelle nous enseigne, quand on ne sait pas répondre, à ne pas répondre. On peut toutefois dire ceci : si un jour nous atteignons au pouvoir proprement divin de créer de tels êtres, pourquoi n'aurions-nous pas acquis celui de les créer bienfaisants?

Table des matières

Premier survol

On fait des fiches 7
L'arme la plus puissante de l'homme : le « codage » 8
Du cri à la mémoire 10
On traite l'information 11
Au cœur de tout : le « programme » . 14
Voici ce que fait l'ordinateur 16
Que veut-on dire quand on dit qu'une machine « comprend » une information ? 16
L'ordinateur est un esclave 21
Articulation de l'ouvrage 22

Première partie

Le software

Chapitre 1 : **Le codage**

Le principe des impulsions binaires . 27
Codage des nombres 33
Mots et caractères 36
Codage des lettres et signes 39
Les instructions et leur codage
Première idée 39
Deuxième idée 41
La valse des adresses 42
L'ossature fondamentale d'un ordinateur 44
Un peu de genèse 46

Chapitre 2 : **L'art des programmes**

Maman prépare un gâteau feuilleté . 49
L'ordre de branchement 51
Vous vous organisez pour extraire une racine carrée 54
Vous programmez l'extraction de la racine carrée 58
Les sous-programmes 64
La monnaie intellectuelle de l'économie moderne 66

Chapitre 3 : **Les langages**

Le b-a-ba du début 69
Le principe des langages symboliques 70
Premier type de langage symbolique : le langage dit « d'assembleur » . . 72
Les langages symboliques évolués . . 75
L'ordinateur renouvelle la grammaire 78

Deuxième partie

Le hardware

Introduction : **Les grandes divisions du hardware** 85

Chapitre 4 : **La commutation**

Le moyen de commuter 87
Les circuits logiques élémentaires . 91
Incursion dans l'algèbre booléenne . 95
Les circuits logiques 98
Le circuit du branchement conditionnel 102
L'addition 103
Philosophie 107

Chapitre 5 : **La mémorisation**

Le tore de ferrite, pendant magnétique du transistor 109
L'organisation de la mémoire . . . 112
Commentaires technologiques et admiratifs 117
De la ferrite en toutes choses 119
Les « flips-flops » 125
Ce qu'est un « registre » 128
Qu'est-ce qu'un « compteur » ? . . . 129

Chapitre 6 : **Puck** **133**

Chapitre 7 : **Les périphériques**

Les cartes perforées 139
Les bandes perforées 142
Le ruban magnétique 143
Les tambours magnétiques 147
Les disques magnétiques 149
Comment lit-on les résultats ? 150
Les imprimantes 152
Les tables traçantes 156
Le tube cathodique 157
Les consoles de visualisation 159
Le mode conversationnel 161

Chapitre 8 : **Un peu d'histoire et un peu de prospective**

Le passé 165
L'avenir 168

Troisième partie

Les applications

Chapitre 9 : **La gestion**

La tenue des stocks 176
La réservation des places d'avion . . 178

Chapitre 10 : **La documentation**

L'homme étouffant sous son savoir ? . 183
Les mots-clés 184
Vous venez demander une bibliographie 186
Quelques autres méthodes 187
Le crime ne paie pas 189
Philosophie de la documentation automatisée 190

Chapitre 11 : **Les applications « en temps réel »**

L'interception 193
Gérer le radar 195
L'ordinateur dans l'industrie 198
Les capteurs et les actuateurs 202

Chapitre 12 : **Médecine et enseignement**

Abaissement ou élévation du médecin ? 205
L'ordinateur et l'enseignement . . . 208

Chapitre 13 : **Applications scientifiques**

Le mystère des gènes 213
Qu'est-ce qu'une équation ? 214
L'instrument conditionne l'art . . . 217
Modèles et simulation 219
La conception assistée 221
Les sciences nouvelles de l'homme . 224

Chapitre 14 : **L'économie politique de l'ordinateur**

L'industrie et son chiffre d'affaires . 227
La saisie des données 228
Métiers de l'informatique 229
La déesse aux vingt bras 231

Conclusion

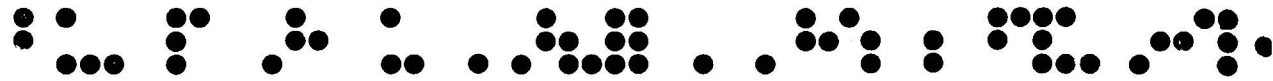
Y-a-t-il révolution ? 237
La vie va-t-elle se robotiser ? 238
Est-on menacé de chômage ? 242
Est-on menacé de tyrannie ? 243
L'ordinateur ravira-t-il à l'homme la couronne de l'intelligence ? 248

BERGER-LEVRAULT, NANCY. — 779933-10-73

BORDAS
INITIATION
collection
dirigée par
Edouard Labin

La science, qui jusqu'ici a éclairé l'homme et accru ses certitudes, commence aujourd'hui à l'angoisser en devenant impénétrable. Personne ne peut plus suivre et dominer le foisonnement de ses prodiges. Et le public, même cultivé, connaît bien mieux les artisans de la culture ancienne que les hiéroglyphes dans lesquels s'inscrit notre avenir.
La littérature dite « de vulgarisation » essaie bien d'éclairer pour tous ce monde en devenir, mais, prisonnière de la vieille pédagogie, elle oscille trop souvent entre la lourdeur et le simplisme.
La collection Bordas-Initiation ambitionne de sortir de cette ornière. Plutôt que de présenter les « produits finis » de la science, elle s'attache à exposer leur genèse et à expliquer le lien entre les questions profondes que se posent les chercheurs et les conquêtes remarquables qu'en tire la technique.
Ni guides pour techniciens, ni mémentos pour étudiants, les volumes de la collection seront plutôt des itinéraires commentés, d'une lecture agréable encore qu'elle doive être attentive, au long desquels on découvrira la science vivante avec son univers de concepts et d'applications.
Ces ouvrages peuvent servir d'introduction à des études plus poussées.
Bordas-Initiation entend couvrir les domaines classiques aussi bien que les secteurs de pointe de la curiosité intellectuelle.

Atelier Pascal Vercken



Cette collection se propose de rendre les sciences et les techniques modernes intelligibles à l'homme cultivé qui ne veut pas être exilé dans son siècle.

L'ordinateur, puis le robot, vont-ils remplacer l'homme et tarir le suc de la vie ? Cette crainte est largement répandue dans le public parce qu'on lui donne sur cette merveilleuse conquête des idées fausses.
Le présent livre est avant toute une démystification. Il fait comprendre en profondeur, et cependant sans lourde technicité, ce qu'est, ce que fait, comment calcule, comment s'emploie un ordinateur, et qu'il n'y a jamais eu d'esclave mécanique plus complètement soumis à son maître humain.
Le lecteur se convainc peu à peu que le seul rôle de l'informatique est d'exécuter, avec des données rassemblées par des hommes et en vue de résultats utilisés par des hommes, des programmes conçus par des hommes, mais dix millions de fois plus vite que ne pourraient le faire des hommes.
Le livre passe en revue les notions essentielles de codage, d'instructions et de programmes, d'opération et de mémoire. On voit ensuite clairement et simplement comment des circuits électroniques peuvent calculer sur des données et les stocker en mémoire, comment on peut introduire en mémoire le programme lui-même écrit dans des langages convenables, et comment l'extérieur de l'ordinateur peut communiquer avec son intérieur et vice versa par des « interfaces » supérieurement ingénieux et puissants — cartes, bandes, disques, imprimantes, écrans cathodiques.
Vient ensuite une revue vivante des applications à tous les domaines, puis quelques chiffres sur l'étonnante explosion de cette industrie et, pour finir, une discussion du thème des thèmes : l'ordinateur pourra-t-il acquérir une intelligence du type et du niveau de celle de l'homme ?

Prochains thèmes traités dans la collection : informatique, électronique, pédagogie, finances et assurances, le commerce international, les systèmes d'armes, les drogues, etc.

I.S.B.N. 2-04-000025-9